VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Enterprise software pro správu a nasazení služeb a aplikací v cloud prostředí

Enterprise Software for Management and Deployment of Cloud Services and Applications

Bc. Lukáš Hanusek

Diplomová práce

Vedoucí práce: Ing. David Ježek, Ph.D.

Ostrava, 2021

Abstrakt

Cílem této práce je vytvořit a popsat software, který bude sloužit jako kompletní řešení pro poskytování cloud hosting služeb s podporou IaaS (Infrastructure as a Service) a PaaS (Platform as a Service) služeb. Je kladen velký důraz za modularitu, tak aby výsledný systém dokázal ovládat široké spektrum aplikací či služeb pomocí webového ovládacího panelu. Způsoby virtualizace jsou rovněž řešeny modulárně, systém je připraven pro podporu několika virtualizačních řešení najednou. Je možné připojit neomezené množství fyzických hostitelských serverů, na které budou služby nasazovány. Pro koncové uživatele bude nasazení aplikace do cloud prostředí záležitost několika kliknutí myší.

Klíčová slova

Cloud, virtualizace, hosting, IaaS, PaaS, web, ovládací panel, nasazování aplikací, Java, Spring, OSGi, LXC, síťování, SSH, FTP, zálohování, obrazy disků, plánování úloh

Abstract

The objective of this thesis is to create and describe software that can be used as a complete solution for a company providing cloud hosting services such as IaaS (Infrastructure as a Service) and PaaS (Platform as a Service). The implementation of this system is highly modular allowing support to deploy many different applications on various platforms. The system is ready to support multiple virtualization technologies using build-in plugin system. It is possible to connect unlimited number of physical servers to deploy applications on. The system is build to be user friendly and deploying applications is done by a few clicks in the web graphical user interface.

Keywords

Cloud, virtualization, hosting, IaaS, PaaS, web, control panel, application deployment, Java, Spring, OSGi, LXC, networking, SSH, FTP, backups, snapshots, scheduling

Poděkování

Rád bych zde poděkoval vedoucímu této Diplomové práce, kterým je Ing. David Ježek, Ph.D. za jeho cenné rady během vývoje projektu a také psaní této práce.

Obsah

Seznam tabulek **6**

Seznam obrázků **7**

1 Úvod **8**

1.1 Cloud modely 8
1.2 Veřejný a soukromý cloud 10
1.3 Stručná přehled poskytovatelů cloud služeb 10
1.4 SLA (Service Level Agreement) 12

2 Cíle praktické části práce **13**

2.1 Požadavky na systém 13
2.2 Motivace 14
2.3 Ustanovení termínů 14
2.4 Role 16
2.5 Diagram nasazení 17

3 Implementace daemon komponenty **18**

3.1 Technologie 18
3.2 Diagram modulů 19
3.3 Platformy 22
3.4 Konfigurace nasazování aplikací (PaaS) 24
3.5 Proměnné v konfiguračních souborech 27
3.6 Uživatelem definované konfigurační proměnné 28
3.7 Šablony a verze aplikací 29
3.8 Aktualizace a změny verzí aplikací 30
3.9 Konfigurace virtualizovaných operačních systémů 30
3.10 Limitování hardware prostředků 32
3.11 Zálohování 32

4 Implementace web komponenty **35**

4.1 Technologie 35
4.2 Lokalizace a jazyky 35
4.3 Instalace a první spuštění webové komponenty 35
4.4 Přihlašování a zabezpečení 37

5 Administrace a konfigurace web komponenty **38**
5.1 Datacentra . . . 38
5.2 Připojení daemon komponenty . . . 38
5.3 Přidání platformy . . . 40
5.4 Přidání aplikace . . . 41
5.5 Přidání hardware konfigurace . . . 43
5.6 Vytvoření plánu . . . 44

6 Nasazování a správa cloud služeb **46**
6.1 Vytvoření služby . . . 46
6.2 Instalace aplikací . . . 46
6.3 Monitorování aplikací . . . 47
6.4 Naplánované úlohy . . . 48
6.5 Datový model služby . . . 49
6.6 Správce souborů . . . 50
6.7 Zálohování . . . 51

7 Další funkce webového rozhraní **52**
7.1 Překlady a lokalizace . . . 52
7.2 Měny . . . 52
7.3 Mezinárodní validace daňového identifikačního čísla . . . 52
7.4 Systém zásuvných modulů pro platební brány . . . 53
7.5 Zabudovaný systém podpory . . . 54
7.6 Varovný systém . . . 54

8 Testované aplikace pro PaaS **55**
8.1 Apache 2 . . . 55
8.2 MariaDB / MySQL . . . 55
8.3 TeamSpeak 3 Server . . . 56

9 Závěr **57**
9.1 Plánované funkce . . . 57

Literatura **58**

Přílohy **58**
A Datový model . . . 59
B Zdrojové kódy aplikace . . . 59
C Návod na instalaci . . . 59
D Sestavená daemon komponenta . . . 59

Seznam tabulek

1 SLA 12
2 Slovník termínů 15
3 Uživatelské role 16

Seznam obrázků

1 Přehled cloud modelů 9
2 OVH Datacentrum v plamenech 12
3 Diagram nasazení 17
4 Diagram modulů 19
5 Struktura dat v repozitáři šablon aplikací 29
6 Nástroj "lxc snapshot"pro vytváření obrazů disků 34
7 První krok instalace - údaje k databázi 36
8 Druhý krok - údaje k SMTP serveru (nevyžadováno) 36
9 Třetí krok - vytvoření prvního účtu administrátora 36
10 Přihlášení do systému 37
11 Přehled existujících datacenter a formulář pro přidání dalšího datacentera 38
12 Rozhraní pro připojení hostitelského serveru - daemon komponenty 39
13 Přidání platformy na straně webu 40
14 Přidání aplikace na straně webu 42
15 Přidání nové konfigurace 43
16 Vytvoření nového plánu 44
17 Část datového modelu zobrazující plány 45
18 Instalace virtuálního operačního systému v reálném čase 46
19 Ukázka instalace nové aplikace 47
20 Monitorování aplikací v reálném čase 47
21 Rozhraní pro vytvoření nové úlohy 48
22 Datový model služby 49
23 Správce souborů - procházení adresářové struktury 50
24 Správce souborů - editor 50
25 Sekvenční diagram zpracování plateb 53
26 "Živá"konzole 56

1 Úvod

Cloud technologie v současné době hrají významnou roli v sektoru služeb i průmyslu, stále více firem digitalizuje své procesy a k tomu potřebují výpočetní prostor dostupný odkudkoliv a ideálně 24 hodin denně. Provozovat svůj vlastní firemní server může být však velmi nákladné, zahrnuje to nejen nákup samotného hardware, ale také najmutí správce. Firemní server a síť je třeba nastavit a zabezpečit. Dále je nutné vyřešit zálohování, důležitá firemní data by měla být zálohována na jiné fyzické umístění, aby v případě živelné katastrofy nedošlo ke ztrátě všech dat. Pokud firma nemá více poboček, může to být obtížně realizovatelné.

Pomocí cloud technologií se dají všechny tyto problémy vyřešit snadno a levněji. Podstatnou výhodou je také škálovatelnost, která umožní dynamicky navýšit výpočetní kapacity v případě, že bude potřeba větší hardware výkon. Naopak, pokud již dodatečný výkon nebude potřeba, můžeme cloud službu nebo její část zrušit a snížit provozní náklady.

1.1 Cloud modely

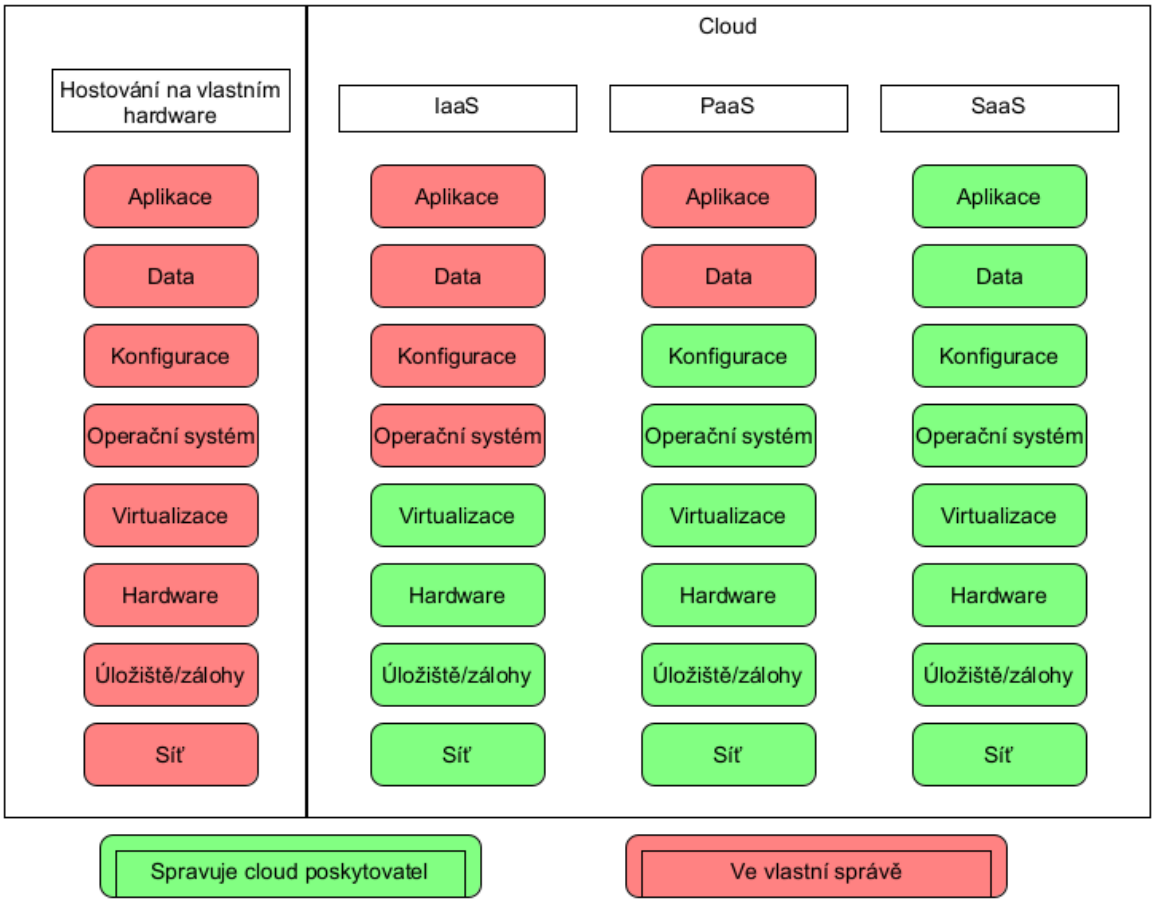
Cloud služby se dají kategorizovat do 3 hlavních modelů. Modely se liší podle rozdělení správy jednotlivých prostředků mezi poskytovatele služeb a samotného zákazníka či koncového uživatele. Rozdělení na modely zobrazuje obrázek 1.

První sloupec na obrázku 1 zobrazuje případ, kdy si firma hostuje všechny své aplikace **na vlastním hardware**, typicky umístěný v prostorách firmy. Firma tak potřebuje vlastní IT oddělení, případně si musí najmout externí firmu na správu serverů. Následující 3 sloupce již reprezentují cloud modely **IaaS**, **PaaS** a **SaaS**.

IaaS (Infrastructure as a Service) je typ služby, která dává uživateli největší svobodu, ale také největší zodpovědnost. Jedná se o virtualizovaný výpočetní prostor. Koncový uživatel si vybere operační systém, ke kterému dostane plný přístup. Nastavení systému, aplikace a data si uživatel spravuje sám. Poskytovatel tohoto modelu má na starosti správu hardware, virtuálního stroje, úložiště, záloh (typicky ve formě obrazů virtuálního disku, označuje se anglickým pojmem *snapshots*) a konektivitu do sítě Internet.

Jako příklad *IaaS* můžeme uvést službu nazývanou jako *VPS* (Virtual Private Server, v překladu: Virtuální soukromý server). Někteří poskytovatelé nabízí také *VDS* (Virtual Dedicated Server, přeloženo jako Virtuální dedikovaný server), jedná se o stejnou službu jako *VPS*, nicméně *VDS* zaručuje, že koncový uživatel dostane garantovaný výpočetní výkon. U služeb *VPS* je časté, že jedno fyzické jádro procesoru je agregováno mezi více virtuálních strojů najednou a výkon tak může kolísat.

PaaS (Platform as a Service) je typ služby, kde si koncový uživatel již nevybírá operační systém, ale je mu přidělen a nakonfigurován podle toho, jaké aplikace chce uživatel provozovat. Uživatel má přístup k souborům a konfiguraci aplikace, ale poskytovatel může některé funkce předem nastavit nebo aplikaci automaticky nainstalovat.



Obrázek 1: Přehled cloud modelů

Příkladem *PaaS* je hostování webových stránek, uživatel má v tomto případě přístup k nahrání vlastního obsahu na web server, samotné nastavení web serveru spravuje výhradně poskytovatel. Mnoho poskytovatelů nabízí automatizovanou instalaci nejpopulárnějších web aplikací, příklady: *WordPress, PrestaShop, Joomla, Drupal, MediaWiki.*

SaaS (Software as a Service) je typ služby, kde koncový uživatel již nevystupuje v roli správce či administrátora služby, ale pouze v roli běžného uživatele. Veškeré nastavení a data aplikace v tomto případě spravuje poskytovatel služby.

Jako zástupce *SaaS* můžeme uvést: *Google Dokumenty*, *Office.com* nebo populární datové úložiště *Google Disk* a *Dropbox.com.*

1.2 Veřejný a soukromý cloud

Cloud technologie lze dělit na soukromé a veřejné cloud služby.

Soukromý cloud je provozován pouze pro účely jedné společnosti nebo organizace, která si spravuje infrastrukturu a software vybavení cloudu sama. Do soukromého cloudu nasazují aplikace jen oprávnění administrátoři z dané organizace. Soukromý cloud může být fyzicky umístěn přímo v firemních prostorech nebo na pronajatém hardware v datacentru.

Veřejný cloud je spravován poskytovatelem, koncový uživatel nebo zákazník se tak nemusí starat o použitý hardware ani software, kterým je cloud infrastruktura tvořena. Stejný hardware a stejnou síť může sdílet několik zákazníků v jednu chvíli najednou, k bezpečnému oddělení služeb jednotlivých uživatelů je použita virtualizace. Provozní náklady na veřejný cloud jsou značně nižší než v případě soukromého cloud řešení.

Seznam několika poskytovatelů cloud řešení je uveden v následující kapitole 1.3.

1.3 Stručná přehled poskytovatelů cloud služeb

V této kapitole zmíním několik poskytovatelů cloud služeb a jejich stručnou historii. Poskytovatelů existuje dnes opravdu hodně, proto jsem vybral pouze některé z největších a nejznámějších (*Amazon Web Services, Google Cloud Platform a Microsoft Azure*). Další méně známe poskytovatele jsem vybral na základě mé osobní zkušenosti (*ScaleWay.com, Hetzner Online, OVH*). Všichni zmínění poskytovatelé provozují služby ve svých vlastních datacentrech a nejedná se o přeprodejce.

1.3.1 Amazon Web Services (AWS)

Zlomový okamžik v historii cloud technologií nastal v Září 2006, kdy společnost Amazon spustila první veřejnou cloud službu s názvem *Elastic Compute Cloud (EC2)*. Tato služba umožní uživatelům vytvořit virtuální stroj během několika vteřin pomocí webového ovládacího panelu nebo *REST API (Application Programming Interface)*. Virtuální stroj je možné zapínat nebo vypínat podle potřeby a pokud už není potřeba, může být kdykoliv odstraněn. Zákazníci jsou účtování podle doby spuštění instance, přenesených dat nebo využitých hardware prostředků. Služba EC2 odpovídá cloud standardu IaaS. [5]

1.3.2 Google Cloud Platform

V dubnu roku 2008 uvedla společnost *Google* produkt **Google Cloud Platform**, podobně jako u služby EC2 od společnosti Amazon, služba odpovídá cloud modelu IaaS. Google na své cloud platformě podporuje také služby typu PaaS, jako například *Cloud SQL*, které interně používají databáze *MySQL*, *PostgreSQL* nebo *Microsoft SQL Server*.

1.3.3 Microsoft Azure

V roce 2010 vstoupila na pole cloud technologií společnost Microsoft se službou **Windows Azure**, krátce na to byla služba přejmenována na *Microsoft Azure*. Tato služba taktéž podporuje vytváření virtuálních strojů na požádání, oproti Amazon EC2 byla ale z počátku vázána na ostatní produkty společnosti Microsoft jako je *Microsoft SQL Server*, *Internet Information Services* a *Windows*. Jednalo se o služby modelu PaaS. Rostoucí popularita operačního systému Linux a *open-source (aplikace s veřejným zdrojovým kódem)* aplikací jako *Apache 2, MySQL a PHP*, učinila Microsoft změnit strategii a přidat podporu pro služby typu IaaS, to umožnilo zákazníkům využít operační systém Linux a open-source aplikace na platformě *Microsoft Azure*.

1.3.4 Scaleway.com

Z dalších poskytovatelů stojí za zmínku také **Scaleway.com**, původně projekt operátorů francouzských datacenter *Online.net* vznikl v roce 2015. Později díky popularitě projektu Scaleway.com došlo k přejmenování celé společnosti z původního názvu *Online.net* na *Scaleway.com*. Popularitu společnosti zajistily, v porovnání s konkurencí, velmi levné služby *IaaS*, které běžely na procesorech *ARM*.

1.3.5 Hetzner Online

Společnost založena původně v roce 1997 se zaměřovala dlouhou dobu pouze na pronájem dedikovaného hardware, hostování webových stránek a kolokační služby (umístění serveru zákazníka do prostoru datacentra) ve svých datacentrech umístěných v sousedním Německu. Na počátku roku 2018 přišel i tento poskytoval s vlastním *IaaS* cloud řešením. [6]

Rád bych také zmínil, že pro testování implementace praktické části této práce byly použity právě dedikované servery této společnosti.

1.3.6 OVHcloud

Cloud technologie nemusí ale nutně znamenat absolutní bezpečnost dat. Bohužel v průběhu psaní této práce se stala tragická událost. Dne 10.3.2021 kompletně shořela budova datacentra označována jako *SBG2* společnosti *OVHcloud*. Část další budovy nesoucí jméno *SBG1* byla také částečně zasažena [7].

Po požáru v datacentru *OVHcloud* umístěném ve Štrasburku (Francie), mnoho zákazníků pomocí sociální sítě *Twitter* nahlásilo kompletní ztrátu dat. Později ve videu, které zveřejnil zakladatel a ředitel společnosti Octave Klaba prostřednictvím sítě *YouTube* potvrdil [8], že většinu dat nebude možné obnovit. Uživatelé, kteří sami neprováděli zálohy svých cloud služeb na jinou fyzickou lokaci tak nenávratně o svá data přišli.

Společnost provozuje ve Francii, Německu, Velké Británii a ve Spojených Státech Amerických několik datacenter. Poskytuje pronájem dedikovaného hardware a také cloud služby, popularitu

Obrázek 2: OVH Datacentrum v plamenech
Zdroj: root.cz (10.3.2021)

tato společnost získala především díky své vysokokapacitní sítě, která dokáže zákazníky chránit před *DDoS (Distributed Denial of Service)* útoky o síle i několika Gb/s.

1.4 SLA (Service Level Agreement)

Zkratka *SLA* zastupuje termín *Service Level Agreement*, který by se dal přeložit jako *dohoda o úrovni služeb*. Tato dohoda je uzavírána mezi poskytovatelem služeb a koncovým zákazníkem. *SLA* dohoda specifikuje poskytované služby a také jejich dostupnost. Dostupnost se uvádí v procentech, které znamenají maximální dobu, po kterou může být služba nedostupná. V případě, že je tato doba přesažena, pak má zákazník právo na náhradu.

Nedostupnost může být způsobena výpadkem elektrické sítě, selháním hardware nebo síťového prvku na straně poskytovatele služby. Typicky se dnes toto číslo pohybuje mezi hodnotami 99.5% až 99.999%.

Následující tabulka 1 udává přehled pro nejpoužívanější SLA hodnoty v oblasti poskytování cloud služeb.

Tabulka 1: SLA

SLA	Maximální nedostupnost za rok	Maximální nedostupnost za měsíc
99%	3.6 dní	7.2 hodin
99.5%	1.8 dní	3.6 hodin
99.9%	8.7 dní	43 minut
99.99%	52 minut	4.3 minut
99.999%	5.2 minut	26 vteřin

2 Cíle praktické části práce

Cílem praktické části této práce je rozšířit a pokračovat ve vývoji *Enterprise* software, jehož vývoj začal v rámci semestrálního projektu v akademickém roce 2019/2020. Software bude sloužit pro správu a nasazování aplikací do cloud prostředí primárně určený pro operační systém Linux a jeho distribuce. Vytvořený software bude podporovat cloud služby typu *IaaS (Infrastructure as a Service) a PaaS (Platform as a Service)*. Systém bude navržen tak, aby modulárně podporoval dnes nejčastěji nasazované aplikace do cloud prostředí, například: *Apache2, PHP, Wordpress, MySQL, MariaDB, Apache Tomcat*. Systém musí být zároveň stavěn tak, aby bylo možné kdykoliv přidat podporu pro nasazení nové aplikace bez nutnosti zasahovat do zdrojových kódů systému.

Systém umožní připojit neomezené množství fyzických hostitelských serverů, pomocí virtualizace na nich budou provozovány cloud služby typu *IaaS a PaaS*.

Systém bude rozdělen na 2 hlavní komponenty, webový ovládací panel dostupný koncovým uživatelům a komponenta, která bude běžet na hostitelských serverech a bude ovládat hostované služby a komunikovat s webovou komponentou. Použitý typ virtualizace bude řešen modulárně, aby systém nebyl závislý pouze na jednom zvoleném typu virtualizace.

Systém umožní koncovým uživatelům spravovat a vytvářet cloud služby pomocí webového grafického uživatelského rozhraní. Výsledkem této práce bude kompletní řešení pro současné potřeby moderního cloud hostingu. Systém musí být v oblasti služeb *PaaS* přívětivý a snadný na ovládání i pro běžné a neznalé uživatele, kteří chtějí provozovat pouze jednoduché webové stránky ale nemají dostatečné znalosti pro instalaci a nastavení webového serveru.

2.1 Požadavky na systém

- Vytváření virtualizované infrastruktury a virtuálních strojů (*IaaS*)
- Podpora pro přidávání různých Linux distribucí pouze pomocí konfigurace (*Debian, CentOS, Fedora, Devuan, Ubuntu* a další)
- Vytváření a nasazování aplikací a služeb do cloud prostředí (*PaaS*)
- Správa sítě, virtuální síťová rozhraní, přiřazování IP adres a portů k jednotlivým službám a virtuálním strojům
- Správa přístupu k souborům služby pro koncové uživatele (*SSH, SFTP, FTP*)
- Živý konzolový výstup aplikací do webového rozhraní a možnost interakce se službou
- Podpora neomezeného počtu fyzických hostitelských serverů pro nasazování služeb
- Modularita podporovaných aplikací
- Modularita virtualizačních technologií

- Monitorování využitých prostředků aplikacemi: využití CPU, paměti, disku, sítě a jejich omezování
- V případě, že aplikace nebude dostupná, systém se pokusí automaticky aplikaci znovu nastartovat, v případě neúspěchu uživateli přijde oznámení
- Zálohování aplikací v definovaném intervalu a možnost kdykoliv aplikaci obnovit do stavu v době zálohy
- Zálohování virtuálních strojů pomocí obrazů disku (*snapshots*) a možnost kdykoliv celý virtuální stroj obnovit do stavu v době, kdy byl obraz disku pořízen
- Naplánování úloh pro virtuální stroj nebo pro některou z instalovaných aplikací (*PaaS*). Úlohy můžou být nastaveny s periodickým opakováním a jejich průběh bude zaznamenán
- Procházení adresářové struktury a souborů dat jednotlivých aplikací pomocí webového rozhraní
- Správa šablon pro jednotlivé aplikace (například *Apache2* web server s předdefinovanou šablonou pro redakční systém *WordPress*)
- Správa uživatelů (registrace, přihlašování, oprávnění) a přístupu k ovládání jednotlivých služeb přes webový ovládací panel

2.2 Motivace

Protože velcí poskytovatelé cloud služeb nemají žádný důvod své software vybavení poskytnout dalším subjektům, vzniká v této oblasti mezera na trhu. Menší nebo začínající společnosti v oblasti cloud hostingu nemají šanci bez systému automatického nasazování služeb v řádu sekund od vytvoření objednávky uspět. Je standardem poskytnout koncovým zákazníkům přehledné webové rozhraní, kompatibilní pro všechny zařízení, tak aby bylo možné cloud služby vytvářet a kontrolovat odkudkoliv.

Praktická část této práce by měla tuto mezeru na trhu vyplnit a poskytnout tak menším firmám dostupné *PaaS* a *IaaS* řešení, které je přizpůsobitelné a konfigurovatelné na všech úrovních.

2.3 Ustanovení termínů

Pro ustanovení termínů použitých pro popis praktické části a implementace této práce slouží následující slovník.

Tabulka 2: Slovník termínů

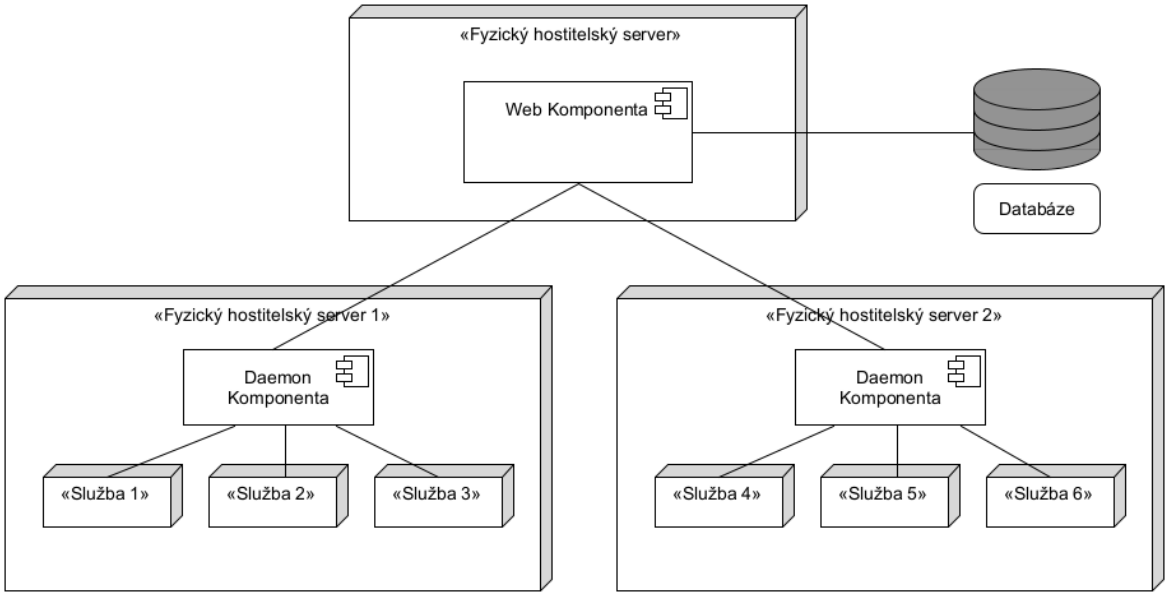
Pojem	Popis
Aplikace	Aplikace / služba spouštěna systémem (*Apache2, MySQL, Tomcat*)
API	Application Programming Interface (rozhraní pro programování aplikací)
Konfigurace	Definuje hardware omezení pro platformy. Limituje procesor, operační paměť, místo na disku a přenosy po síti.
Komponenta	Systém je rozdělen na 2 komponenty, které mezi sebou komunikují: daemon a web.
Daemon	Komponenta systému bežící na fyzických hostitelských serverech. Ovládá služby a zajišťuje jejich spouštění a ovládání, předává informace webové komponentě. V typickém použití bude několik daemon komponent připojených k jedné instanci webové komponenty.
IaaS	Infrastructure as a Service
LXC	LinuX Containers - Open-source virtualizační platforma na úrovni OS pro Linux systémy.
Plán	Definuje platformu, šablonu a konfiguraci pro vytváření služeb
Platforma	Prostředí, které definuje jak a kde se aplikace spouští.
Server	Fyzický hardware, na kterém je realizována virtualizace
Šablona	Připravené soubory v repozitáři pro instalaci aplikace
Služba	Základní organizační jednotka. Definuje výpočetní prostor, kde je možné instalovat a spouštět aplikace
PaaS	Platform as a Service
Verze	Každá aplikace může mít několik verzí. Například pro *Apache2* web server: *Apache2 + PHP 7.2 nebo PHP 7.3*
Veřejná IP adresa	IP plně dedikovaná pro službu, nevyužívá *NAT* ani žádnou formu síťového mostu pro přemostění jen určitých portů, všechny TCP/UDP porty jsou dostupné ze sítě Internet

2.4 Role

Tabulka 3: Uživatelské role

Role	**Popis**
Administrátor (admin)	Má plný přístup ke všem službám, včetně plného přístupu do administrace
Zákaznická podpora	Přístup pouze k základním informacím o službách ostatních uživatelů, má plný přístup do zabudovaného *help-desk* systému
Majitel služby / koncový uživatel	Má přístup pouze k službám, které vlastní
Uživatel	Registrovaný uživatel, nemá žádná práva, dokud mu není přidělena služba nebo oprávnění ke službě

2.5 Diagram nasazení



Obrázek 3: Diagram nasazení

Diagram na obrázku 3 zachycuje jeden ze způsobů možného nasazení vytvořeného systému v rámci této práce. Web komponenta umístěná na jednom fyzickém hostitelském serveru je připojena k databázi a ke dvěma daemon komponentám, každá z nich běží na jiném fyzickém hostitelském serveru a spravuje jiné služby. Pojmem služba je zde myšlen model cloud služeb *IaaS* nebo *PaaS*.

3 Implementace daemon komponenty

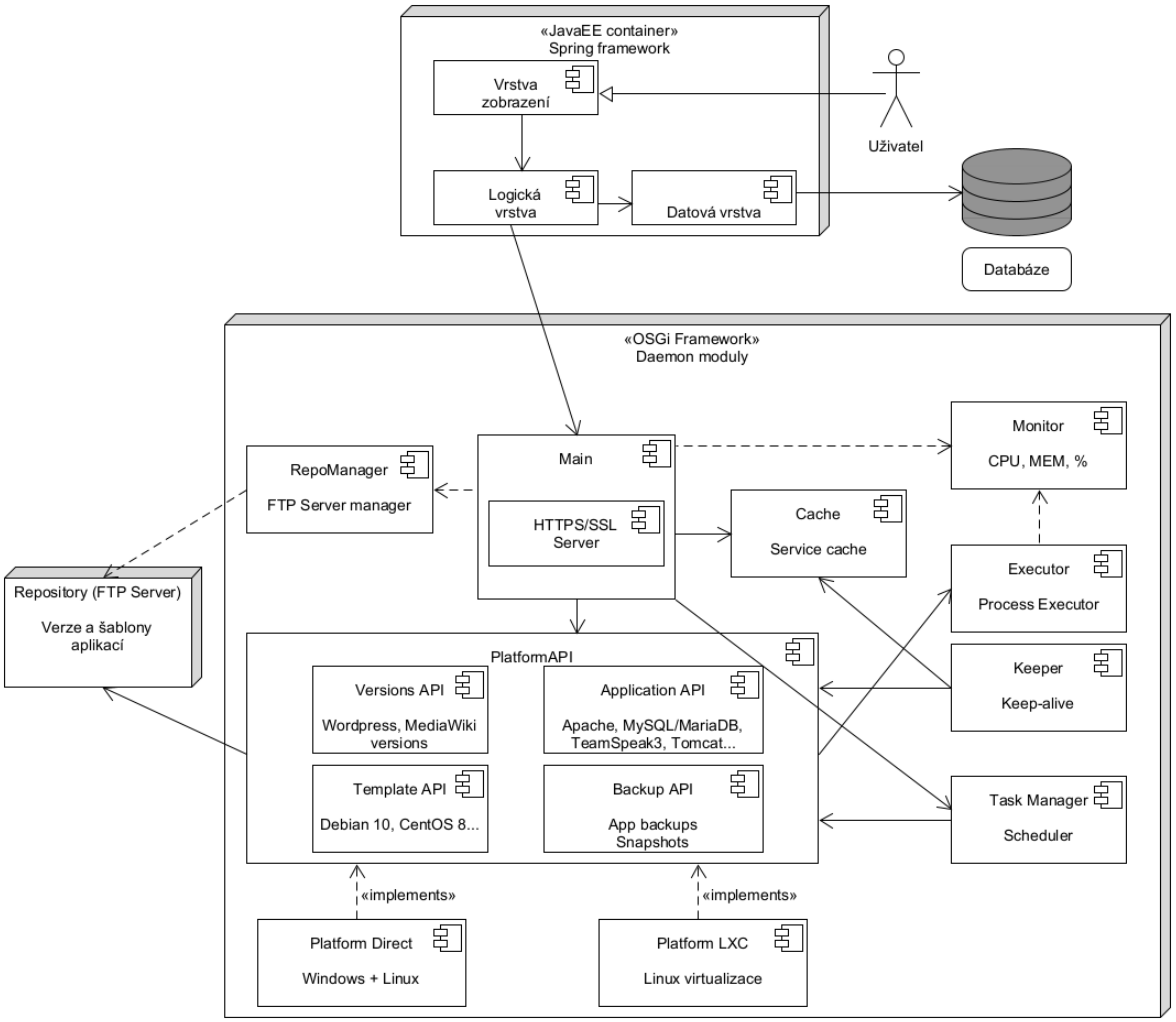
Daemon je software komponenta, která je nasazena na fyzických hostitelských serverech. Podle konfigurace zajišťuje nasazování, ovládání a chod cloud služeb *PaaS* a *IaaS*. Daemon podporuje operační systémy Windows a Linux. Na systémech Windows není podporována virtualizace. Pro spuštění komponenty je třeba mít na operačním systému nainstalovanou Java 9 nebo vyšší a správně nastavenou systémovou proměnnou *JAVA_HOME*.

3.1 Technologie

Daemon je postaven na Java technologii *OSGi* [4], tím je umožněno rozdělit software do modulů (OSGi bundle), které je v případě potřeby možné za běhu přidávat nebo aktualizovat. To umožní přidat další funkce (moduly), aniž by došlo k přerušení služeb pro koncové uživatele. Také je možné moduly aktualizovat nebo restartovat bez přerušení služeb. Zde platí pouze jedna výjimka, jedná se modul, který slouží pro uchovávání informací o spuštěných procesech, nicméně modul je stavěn tak, aby i přes větší aktualizace a úpravy systému, nebylo třeba tento modul nijak modifikovat.

Další z výhod, která plyne z *OSGi frameworku* je samotná modularita, která pomáhá snižovat komplexitu a umožňuje rozšíření systému o další moduly a funkce díky uspořádání do modulů s jasně definovaným komunikačním rozhraním.

3.2 Diagram modulů



Obrázek 4: Diagram modulů

Diagram nasazení zobrazuje jednotlivé *OSGi* moduly (v anglické terminologii *OSGi bundle*) a provázaní mezi nimi. Směr šipky v diagramu udává směr komunikace a závislosti mezi moduly. Je zde zobrazena i webová komponenta, aby bylo patrné, kde dochází k výměně dat mezi těmito komponentami. Web komponenta je popsána detailně v kapitole 4. Následující podkapitoly se zaměřují na popis funkcí jednotlivých modulu na straně daemon komponenty.

3.2.1 Modul Main

Modul **Main** implementuje některé příkazy do *OSGi* konzole (znovu načtení konfiguračních souborů a příkazy pro lokální testování). Zároveň tento modul obsahuje *HTTPS/SSL* server, který zpracovává požadavky z webové komponenty a umožňuje tak, aby byl daemon ovládatelný vzdáleně.

Daemon vždy vystupuje v roli serveru, přijímá požadavky od web komponenty a odpovídá na ně, sám ale žádné neposílá, díky této vlastnosti je možné nasadit několik instancí web komponenty distribuovaně. Tělo požadavku je tvořeno sekvencí bajtů, pro serializaci objektů je použit *ObjectInputStream* a *ObjectOutputStream* z balíku *java.io*. Server aktuálně podporuje 46 příkazů, mezi ně patří získaní stavu služeb a aplikací, zastavení nebo nastartování služby, informace o zálohách, instalace a nasazování aplikací a další. Server je zabezpečen komunikačním heslem, které se posílá společně s každým požadavkem a šifrování protokolu *HTTPS* pomocí *SSL*.

3.2.2 Modul Cache

Tento modul uchovává poslední verzi informací, které na instanci dorazily ze strany web komponenty (4). Protože samotná daemon komponenta není připojená k databázi, je závislá pouze na informacích, které obdrží z web komponenty. Všechny obdržené informace jsou drženy v tomto modulu. Pokud by web komponenta byla nedostupná, daemon stále může na základě uložených údajů v tomto modulu operovat samostatně a provozované aplikace nebudou výpadkem webu nijak ovlivněny.

3.2.3 Modul Keeper

Tento modul je přímo závislý na modulu **Cache**, kontroluje stav všech provozovaných aplikací a v případě, že dojde k pádu aplikace, která má nastaven automatický start v případě pádu, modul odešle požadavek na znovu nastartování aplikace přes modul **PlatformAPI**. V případě pádu několika aplikací najednou, bude použita fronta a aplikace se postupně spustí s časovým rozestupem, aby nedošlo k přetížení systému. Pokud se aplikaci nepodaří nastartovat, bude umístěna dočasně na černou listinu, aby neblokovala frontu pro další aplikace čekající na automatický start po pádu.

3.2.4 Modul PlatformAPI

PlatformAPI definuje rozhraní (API) pro platformní moduly, které se starají o spouštění a ovládání služeb na jednotlivých platformách. Je možné kdykoliv přidat další platformy, bez nutnosti zasazovat do existujících modulů. Platformou je myšleno prostředí, ve kterém se vytváří a spouští služby *IaaS* a *PaaS*. Prostředím může být například virtuální stroj s operačním systémem Linux.

Pomocí **PlatformAPI** je možné do systému v budoucnu implementovat podporu pro další virtualizační technologie (*KVM, Xen, Qemu* a další), systém tak není závislý pouze na jednom virtualizačním řešení a může jich využívat i několik najednou.

PlatformAPI je rozděleno do 4 hlavních rozhraní, které musí každý platformní modul implementovat, aby bylo možné modul načíst a nasazovat přes něj služby:

- **Application API** definuje rozhraní pro spouštění jednotlivých aplikací pro služby typu PaaS. Přes toto rozhraní je možné předávat příkazy do modulu *Executor*, který se stará o samotné spouštění aplikací.
- **Template API** definuje rozhraní pro vytváření služeb a pro práci s šablonami operačních systémů.
- **Versions API** definuje rozhraní pro práci s verzemi aplikací, detailnější popis použití tohoto rozhraní je popsán v sekci 3.7).
- **Backup API** poskytuje rozhraní pro implementaci zálohování na platformách. Může se jednat o zálohy souborů jednotlivých nainstalovaných aplikací nebo obrazy disků celých virtuálních strojů (*snapshots*).

3.2.5 Modul Executor

Executor modul spouští a uchovává informace o všech spuštěných aplikacích. Pokud se jedná o aplikaci s konzolovým vstupem/výstupem, pak tento modul zachytává a ukládá výstup, ten je poté předán a zobrazen přes web komponentu. **Executor** modul dále volitelně předává (záleží na typu aplikace či služby) informace o procesu do modulu **Monitor**.

3.2.6 Modul Monitor

Modul monitoruje hardware prostředky, které daný proces momentálně využívá - využití procesoru, paměti, disku a sítě. Modul používá multiplatformní knihovnu *OSHI* [14] a nové *Java 9+ Process API* [15] pro čtení hodnot o využití hardware prostředků z operačního systému. Modul dále monitoruje stav a globální využití prostředků (zátěž procesoru, obsazenost paměti RAM a na operačních systémech Linux je monitorován i systémový *load* za 1, 5 a 15 minut).

3.2.7 Modul RepoManager

Volitelný modul, který nainstaluje a nastartuje *FTP server Apache MINA* (chráněný heslem). Tento FTP server lze poté registrovat do všech instancí daemon komponent a všechny tak budou sdílet jeden repozitář se všemi šablonami pro aplikace. Tato centralizace tak usnadní správu administrátorům, v případě provádění změn v šablonách. Tento modul je volitelný, a není třeba ho použít, šablony mohou být umístěny na jakémkoliv FTP serveru. Účel tohoto

modulu je zrychlit a automatizovat instalační proces, administrátor tak nemusí instalovat FTP software ručně, pokud nemá k dispozici existující FTP server.

3.2.8 Modul TaskManager

Modul slouží k spouštění naplánovaných úloh. Úlohy mohou být naplánovány na konkretní datum a čas s přesností na vteřiny a možností nastavení intervalu opakování.

Typy podporovaných úloh:

- **Start platformy** - Spustí platformu, v případě platformy LXC (3.3.2) se jedná o virtuální stroj
- **Zastavení platformy** - Zastaví platformu (virtuální stroj)
- **Start aplikace** - Spustí nainstalovanou aplikaci, v případě, že je platforma zastavena, úloha nejdříve spustí platformu a poté až nainstalovanou aplikaci
- **Zastavení aplikace** - Opak příkazu Start aplikace, zastaví se pouze aplikace, platforma (virtuální stroj) nebude zastavena
- **Záloha** - Pro zadanou aplikaci spustí zálohu (detaily zálohování jsou popsány v kapitole 3.11)
- **Vstup** - Úloha odešle do aplikace s aktivním vstupně-výstupním proudem zadaný vstup (typicky se bude jednat o příkaz do konzolového rozhraní spuštěné aplikace)

3.3 Platformy

Modul **PlatformAPI** umožňuje načtení i několika modulů platforem na jedné instanci daemon komponenty současně, takže způsob virtualizace a použité virtualizační řešení se může pro každou službu lišit. V případě potřeby je možné v budoucnosti systém velmi snadno a rychle rozšířit o další virtualizační řešení jako třeba *KVM* nebo *Xen*.

3.3.1 Platforma Direct

Tato platforma spouští aplikace na stejném operačním systému jako běží daemon instance a nevyužívá žádnou virtualizaci. Podporuje operační systémy Windows a Linux. Na systémech Linux navíc umožňuje spouštět aplikace a služby pod oddělenými uživateli pro zvýšení bezpečnosti a také to umožňuje použít uživatelské účty pro přístup k souborům aplikace pomocí protokolu FTP. Tato platforma však nenabízí žádnou možnost jak limitovat hardwarové prostředky, které může služba využít.

Možné využití této platformy může být například web hosting často označován jako *LAMP (Linux, Apache2, MariaDB/MySQL a PHP)*. Několik uživatelů může sdílet stejný databázový

server s oddělenými databázemi, stejný web server s odděleným prostorem pro data pro každého uživatele a stejnou verzi *PHP*.

Další možnost jak využít tuto platformu je použití pouze jako soukromý cloud, kdy na tuto platformu nasazují aplikace pouze oprávněni administrátoři a není potřeba služby od sebe oddělovat a virtualizovat. Jako příklad je možné zmínit oblast herního hostingu. Provozovatelé online video her potřebují často hostovat stovky instancí stejné aplikace (herního serveru), v tomto případě je možné využít tuto platformu pro správu všech instancí a načítat data ze stejného místa, pro nahrání aktualizace tak nemusí aktualizovat data na všech instancích, ale pouze v repozitáři s šablonou a následně všechny instance pouze restartovat.

3.3.2 Platforma LXC

Tato platforma využívá virtualizační technologii *LXC (LinuX Containers)*, což je software s otevřeným zdrojovým kódem a je stále velmi aktivně komunitně vyvíjen. Software *LXC* sloužil v počátcích dnes již rozšířeného software *Docker* jako virtualizační řešení.

Tato platforma instaluje a spouští jednotlivé aplikace (*PaaS*) v oddělených virtuálních kontejnerech, tím je možné limitovat využití procesoru, paměti, disku a sítě. Limitací hardware prostředků je zajištěno, že koncový uživatel vždy dostane přesně přidělené prostředky a nemůže přetížit systém a omezit tak služby ostatních uživatelů (limitování hardware je dále popsáno v sekci 3.10). Protože LXC kontejner neobsahuje emulátor hardware komponent jedná se pouze o virtualizaci na úrovni operačního systému, což poskytuje plný výkon daného hardware. K limitování hardware prostředků software LXC využívá funkci **cgroups** *(control groups)* z jádra operačního systému Linux.

Kontejner může využít veřejné IP adresy (pokud je přidělená) a obsadit jakýkoliv TCP/UDP port. Pokud běží virtuální stroj na sdílené IP adrese jsou mu podle instalovaných služeb přiděleny volné TCP/UDP porty (z definovaného rozsahu, který je nastaven na straně web komponenty, více v kapitole 5.2) pro přístup k virtuálnímu stroji (*FTP, SFTP nebo SSH*) a pro nainstalované aplikace na virtuálním stroji. TCP/UDP porty jsou následně přesměrovány na virtuální stroj pomocí síťového mostu, který je součástí standardní instalace virtualizačního software LXC. Díky zabudované funkci síťového mostu přímo v *LXC* virtualizaci je možné správu sítě jednoduše automatizovat a není potřeba porty přesměrovávat pomocí dalšího software jako *iptables* nebo jiný firewall systém.

Platforma podporuje 2 cloud modely a to **PaaS**, kde konkrétní aplikace je nainstalována pomocí předem připravených konfiguračních souborů a skriptů (popsáno v sekci 3.4.1) přímo do virtuálního stroje. Uživatel nevybírá konkrétní operační systém, ale pouze typ aplikace, které chce nainstalovat a operační systému mu bude přidělen automaticky podle zvolené aplikace. Druhý podporovaný cloud model **IaaS** umožní uživateli vybrat si distribuci operačního systému Linux a uživatel dostane plný přístup k virtuálnímu stroji, instalace aplikací je tak plně v rukou koncového uživatele.

Tato platforma je vhodná pro použití jako veřejný cloud hosting (1.2).

3.4 Konfigurace nasazování aplikací (PaaS)

Podpora pro více aplikací je zajištěna pouze na bázi konfigurace. Každá aplikace má svůj vlastní konfigurační soubor, ve kterém je nutné definovat veškeré operace, potřebné pro manipulaci s danou aplikací. Konfigurace aplikace se vždy váže na konkrétní verzi operačního systému, protože instalace a ostatní operace nad aplikací se mohou na jednotlivých operačních systémech nebo distribucích operačního systému Linux lišit.

Následující podkapitola 3.4.1 popisuje možnosti konfigurace s konkrétní ukázkou konfiguračního souboru ve formátu *YAML*. Pro přidání podpory dalších aplikací není nutné implementovat zásuvný modul nebo jinak modifikovat zdrojový kód některého z modulů daemon komponenty.

3.4.1 Konfigurace aplikace - MariaDB příklad

```
code: MariaDB
name: MySQL/MariaDB for Debian 10
home: "/home/$appCode/$appId"
startCommand: "/etc/init.d/mysql start"
stopType: HOST
stopCommand: "/etc/init.d/mysql stop"
status:
  type: COMMAND
  command: "/etc/init.d/mysql status"
preStartOperations:
- operation: CHANGE_FILE
  file: "/etc/mysql/mariadb.conf.d/50-server.cnf"
  changeKey: port
  newValue: "=$port"
- operation: APPEND_LINE_IF_NOT_EXISTS
  file: "/etc/mysql/mariadb.conf.d/50-server.cnf"
  changeKey: port
  newValue: port=$port
- operation: CHANGE_FILE
  file: "/etc/mysql/mariadb.conf.d/50-server.cnf"
  changeKey: bind-address
  newValue: "=#dbbind"
installOperations:
- operation: COMMAND
  command: apt update
- operation: COMMAND
  command: apt -y install mariadb-server
```

```
- operation: COMMAND
  command: mysql -u root -e "ALTER USER 'root'@'localhost' IDENTIFIED BY '#
      dbpassword';"
- operation: COMMAND
  command: mysql -u root -e "DELETE FROM mysql.user WHERE User='';"
- operation: COMMAND
  command: 'mysql -u root -p#dbpassword -e ''create database #dbname'''
postInstallOperations: []
uninstallOperations:
- operation: COMMAND
  command: apt -y purge mariadb-server
preBackupOperations:
- operation: COMMAND
  command: mysqldump --add-drop-database --skip-lock-tables --flush-privileges
      --all-databases -h localhost -u root -p'#dbpassword' > $home/backup.sql
postBackupOperations:
- operation: COMMAND
  command: rm $home/backup.sql
preRestoreOperations:
- operation: COMMAND
  command: rm -r $home
postRestoreOperations:
- operation: COMMAND
  command: mysql -u root -p'#dbpassword' < $home/backup.sql
backupFolders:
- name: main
  folder: "$home"
  excludedFolders: []
```

3.4.2 Popis konfiguračního souboru pro MariaDB

Konfigurační soubor neobsahuje v této ukázce žádné komentáře, nicméně v ukázkovém konfiguračním souboru (umístěn příloze společně se zdrojovými kódy aplikace) je ke každé konfigurační hodnotě napsána dokumentace včetně všech dostupných systémových proměnných.

- **code** - Unikátní identifikátor pro danou aplikaci, není možné později modifikovat
- **name** - Delší název aplikace
- **startCommand** - Příkaz do terminálu operačního systému pro spuštění aplikace

- **stopType** - Způsob, jakým dojde k zastavení. Protože *MariaDB* běží na pozadí (v terminologii Linuxu - *daemon*, v terminologii Windows - služba), je zde typ *HOST*, což znamená, že příkaz pro zastavení se bude posílat přímo do terminálu operačního systému. Je zde také možné použít typ *PROCESS*, který pošle aplikaci s aktivním konzolovým vstupem a výstupem příkaz pro ukončení (pokud ho daná konzolová aplikace podporuje).

- **stopCommand** - Příkaz do terminálu operačního systému nebo do vstupního proudu aplikace pro zastavení, způsob jakým bude vykonán závisí na předchozím parametru **stopType**

- **status** - Definuje způsob získání stavu aplikace. Položka *type* může nabývat hodnot *COMMAND* nebo *PROCESS*. Typ *COMMAND* vyvolá příkaz do terminálu operačního systému, přečte jeho výstup a podle hodnoty *successString* se vyhodnotí, zda je daná aplikace spuštěná. Typ *PROCESS*, podobně jako u parametru *stopType*, slouží pro aplikace s aktivním vstupně/výstupním proudem, další parametry nejsou třeba a systém pouze ověří, zda je proces aktivní.

- **preStartOperations** - Definuje seznam operací, které se budou vykonávat vždy před samotným spuštěním aplikace. Typické využití je například nastavení TCP/UDP portů nebo limity operační paměti RAM v konfiguračních souborech aplikace, jako je vidět v ukázce s *MariaDB*

  Jsou podporovány následující operace:

  - Změna hodnoty v textovém souboru podle definovaného klíče (*CHANGE_ FILE*)
  - Vytvoření nového souboru (*CREATE_ FILE*)
  - Vložení nového řádku na konec souboru (*APPEND_ LINE*)
  - Vložení nového řádku, pokud již soubor daný řádek neobsahuje (*APPEND_ LINE_ IF_ NOT_ EXISTS*)
  - Smazání řádku (*DELETE_ LINE*)
  - Smazání souboru nebo složky (*DELETE_ FILE*)
  - Provedení příkazu do terminálu operačního systému (*COMMAND*)

  Tyto operace se provádějí vždy sekvenčně, aby bylo možné definovat navazující operace, které vyžadují, aby byla předchozí operace dokončena.

- **installOperations** - Seznam operací se zde definuje stejně jako v předchozím bodu. Operace se provádějí před stažením šablony aplikace (v případě *MariaDB* se šablona nestahuje, protože veškeré instalační soubory jsou staženy automaticky pomocí balíčkového systému *apt* pro *Linux Debian*)

- **postInstallOperations** - Pokud se jedná o aplikaci, kde je potřeba stáhnout instalační soubory, pak použijeme tuto sadu operací k vyvolání instalačních akcí nad staženými soubory ze šablony aplikace
- **preBackupOperations** - Seznam operací, které se provedou před spuštěním zálohy. V *MariaDB* příkladu je definována operace *"mysqldump"*, která exportuje obsah databáze do SQL souboru, ten bude umístěn do adresáře *$/home/backup.sql*
- **postBackupOperations** - Seznam operací, které se spustí po provedení zálohy. Lze je využít k vyčištění již nepotřebných nebo dočasných souborů vzniklých během procesu zálohování.
- **preRestoreOperations** - Tyto operace budou spuštěny před obnovením dat ze zálohy, v případě ukázky pro *MariaDB* se pouze spustí příkaz na promazání domovského adresáře (používá se pouze pro dočasné soubory záloh)
- **postRestoreOperations** - Tyto operace budou spuštěny po obnovením dat ze zálohy, v zmíněné ukázce pro *MariaDB* najdeme operaci pro import SQL souboru do databáze (toto je pouze ukázkový příklad využití této konfigurační položky)
- **backupFolders** - Seznam adresářů, které budou součástí zálohy, pro každý adresář je možné definovat seznam výjimek, které nebudou zahrnuty ve výsledné záloze. Do seznamu výjimek stačí definovat jméno nebo začátek jména adresáře, může se nacházet i v podadresáři. Zálohování více adresářů může být použito pro aplikace, které ukládají konfigurační soubory do jiného umístění, než ostatní datové soubory aplikace. Detaily implementace zálohování jsou dále popsány v kapitole 3.11.

3.5 Proměnné v konfiguračních souborech

V ukázce 3.4.1 bylo použito několik proměnných. Ve všech konfiguračních souborech je možné použít 2 typy proměnných. Prvním typem jsou **Systémové proměnné**, ty vychází z konfigurace systému a samotné platformy, například domovský adresář nebo jméno aplikace a přiřazené TCP/UDP porty. Systémové proměnné se v konfiguračních souborech definují pomocí symbolu dolaru ($) a bez mezery následuje jméno proměnné.

Dostupné systémové proměnné na úrovni aplikace:

- **$serviceId** - unikátní identifikátor služby
- **$appCode** - unikátní identifikátor nainstalované aplikace
- **$ip** - přidělená adresa IP
- **$mem** - přidělená paměť RAM
- **$port** - TCP/UDP port přidělený nainstalované aplikaci

- **$home** - domovský adresář nainstalované aplikace

Dostupné systémové proměnné na úrovni platformy:

- **$nchome** - pracovní adresář instance daemon komponenty
- **$localIp** - lokální IP adresa přidělená službě
- **$cpus** - počet přidělených virtuálních CPU ke službě
- **$cpuLimit** - maximální zatížení procesoru v procentech, které služba může používat
- **$diskSize** - velikost disku v GB
- **$diskReadLimit** - maximální rychlost čtení z disku (Mb/s)
- **$diskWriteLimit** - maximální rychlost zápisu na disk (Mb/s)
- **$user** - uživatel pro vzdálený přístup ke službě (FTP, SFTP, SSH - typ využitého protokolu záleží podle implementace na konkrétní platformě)
- **$password** - heslo uživatele pro vzdálený přístup ke službě

3.6 Uživatelem definované konfigurační proměnné

Druhým typem proměnných jsou **uživatelské proměnné**, vztahují vždy ke konkrétním aplikacím. V ukázce 3.4.1 bylo použito několik uživatelských proměnných, v konfiguračních souborech definují pomocí symbolu hashtag (#) a bez mezery následuje jméno proměnné.

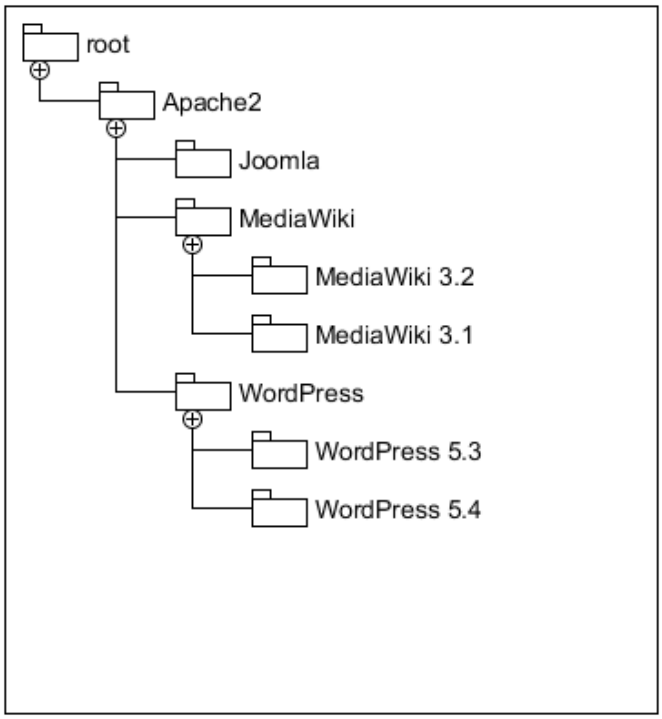
Tento druh proměnných se využívá pouze v službách typu PaaS, které umožňují automatizovanou instalaci aplikací. Pro každou aplikaci je možné definovat neomezený počet proměnných a jejich výchozí hodnoty. Uživatel, který provádí instalaci aplikace, je před instalací požádán o zadání hodnot proměnných, nebo je ponechat na výchozích hodnotách. Hodnoty se mohou použít například pro definici argumentů spustitelného souboru, hesla při instalaci MySQL/MariaDB databáze, nebo pro nastavení velikosti paměti, kterou si může alokovat databáze, to je podstatné v případě, že uživatel bude chtít na platformě provozovat více služeb a nechce veškerou paměť alokovat pro databázi.

Pomocí uživatelem definovaných proměnných je možné plně automatizovat proces instalace aplikace a počáteční nastavení, tím je zajištěno jednoduché a rychlé nasazování aplikací pro koncové uživatele.

3.7 Šablony a verze aplikací

Pro aplikace v rámci služeb PaaS je možné vytvořit připravenou adresářovou strukturu pro konkrétní aplikace. Uživatel při instalaci aplikace může vybrat již předem připravenou verzi. Verze jsou pro každou aplikaci shlukovány do kategorií a v každé kategorii může být několik různých verzí. Verze jsou poté během instalace stahovány pomocí protokolu FTP. Údaje k FTP serveru, kde se šablony nachází je třeba nastavit do konfiguračního souboru.

Jako FTP server je možné využít přiložený Apache MINA FTP server společně s volitelným module RepoManager (3.2.7), který FTP server automaticky předkonfiguruje a spustí společně se spuštěním daemon komponenty.



Obrázek 5: Struktura dat v repozitáři šablon aplikací

V kořenovém adresáři FTP serveru se nachází adresáře *(Apache2)*, jejichž jméno odkazuje na unikátní identifikátor aplikace definovaný v konfiguraci (ukázka konfigurace v sekci 3.4.1), v podsložce se poté nachází podadresáře reprezentující jednotlivé kategorie verzí *(Joomla, MediaWiki, WordPress)*. V adresářích kategorií jsou další adresáře reprezentující konkrétní verze, v těchto adresářích se již nachází data dané aplikace v konkrétní verzi.

3.8 Aktualizace a změny verzí aplikací

Systém umožňuje kdykoliv změnit verzi šablony nebo aktualizovat dříve nainstalovanou aplikaci na novější verzi. K tomu slouží speciální konfigurační soubor *updateOperations.yaml*, který je umístěn v repozitáři společně s daty šablony aplikace, je zde možné definovat operace stejnou syntaxí, jako u konfiguračního souboru aplikací, který byl popsán v kapitole 3.4.2. Operace se zde dají využít například na aktualizaci instalované aplikace, na spuštění skriptu pro konverzi perzistentních dat do nového formátu nebo smazání již nepotřebných souborů.

Ukázka souboru *updateOperations.yaml* s definovanou operací *DELETE_FILE*. Je zde využita proměnná *$home*, která odkazuje na domovský adresář nainstalované aplikace. Definovaná operace smaže adresář */bin* v domovském adresáři před stažením souborů z nové verze z repozitáře.

```
{
updateOperations:
- operation: DELETE_FILE
  file : "$home/bin/"
}
```

3.9 Konfigurace virtualizovaných operačních systémů

Aby bylo možné zajistit podporu pro několik virtualizovaných operačních systémů pro služby *IaaS* a *PaaS* je potřeba každý podporovaný systém nakonfigurovat odděleně, protože se instalace jednotlivých aplikací bude na odlišných operačních systémech lišit. Systém je postaven tak, aby pro přidání podpory dalších distribucí Linuxu stačilo pouze přidat další konfigurační soubor, který obsahuje skripty pro základní nastavení systému a především pro nastavení síťového rozhraní.

3.9.1 Ukázka konfigurace operačního systému Debian 10:

```
code: "Debian10"
name: "Debian 10"
replaceNetConfig: true
networkSetupOps: []
networkPostSetupOps: []
netConfigPath: "/etc/network/interfaces"
netInterface: "eth0"
netConfig:
- "auto lo"
- "iface lo inet loopback"
- ""
- "auto $netInterface"
- "iface $netInterface inet static"
- "    $ip/32"
- "source /etc/network/interfaces.d/*"
postInstallOps:
- IFUP
- INSTALL_SSH
availableOps:
- name: IFUP
  steps:
    - "ifup $netInterface"
- name: INSTALL_SSH
  steps:
    - "apt update"
    - "apt -y install ssh"
    - "sh -c \"echo 'PermitRootLogin yes' >> /etc/ssh/sshd_config\""
    - "/etc/init.d/ssh restart"
```

Konfigurace pro operační systémy umožňuje definovat sadu operací *availableOps*, tyto operace se skládají z jednotlivých kroků, lze také definovat operace, které se spustí hned po úspěšném nainstalování operačního systému a následného prvního spuštění, tyto operace jsou definovány v seznamu (*postInstallOps*). V ukázkové konfiguraci pro *Debian 10* je nastaveno, aby se po instalaci vykonaly operace: *IFUP a INSTALL_SSH*. První z nich spustí síťové rozhraní, které je nutné pro další operace. Druhá definovaná operace stáhne z repozitářů balíčkového systému *apt SSH* server.

3.10 Limitování hardware prostředků

Každá vytvořená služba má ve svém plánu přidělenou také hardware konfiguraci, která definuje limity pro využití hardware prostředků, samotné vymezení pak probíhá při vytváření nových virtuálních strojů. Vytváření hardware konfigurací je dostupné přes web panel, dále popsáno v kapitole 5.5.

Seznam prostředků, které lze limitovat:

- **počet jader procesoru** - limituje počet logických (nikoliv fyzických) jader procesoru, které může virtuální stroj využít
- **maximální využití procesoru v %** - udává maximální zátěž procesoru v procentech, kde 100% odpovídá využití celého jednoho jádra a 200% využití 2 jader. Tento systém přepočtu na procenta využívá většina známých monitorovacích nástrojů v operačním systému *Linux* (*top, htop* a další).
- **operační paměť** - limit operační paměti, kterou může virtuální stroj alokovat
- **velikost disku** - maximální velikost virtuálního disku
- **limit čtení** - limituje maximální rychlost čtení z disku
- **limit zápisu** - limituje maximální rychlost zápisu na disk
- **limit sítě** - omezení přenosové rychlosti přes síťové rozhraní

3.11 Zálohování

Zálohování je implementováno pomocí *PlatformAPI* (3.2.4) na každém platformním modulu odděleně, pomocí *PlatformAPI* je zajištěno, že s oběma platformami lze komunikovat jednotným způsobem. Zálohování se dále dělí na zálohování dat aplikací a vytváření obrazů celého disku virtuální strojů (často označováno anglickým pojmem - *snapshot*).

3.11.1 Zálohování aplikací

Každá aplikace má v rámci svého konfiguračního souboru definován seznam adresářů, které je třeba zálohovat, také je možno upravit chování záloh pomocí operací, které jsou spuštěn před a po vytvoření zálohy (ukázka v sekci 3.4.1). Způsob jakým dojde k záloze definovaných adresářů je určen konfigurací daného platformního modulu, společně s místem konečného uložení a způsobem obnovení dat.

Část z konfiguračního souboru platformy *LXC* (3.3.2) zobrazující nastavení zálohování aplikací (pro služby z modelu *PaaS*) na této platformě:

```
...
dirPath: "/var/snap/lxd/common/lxd/storage-pools/$lxcDiskPool/containers/"
backupCommand: "tar -zcvf $name.tar.gz -C $folder . $excludedFolders"
backupExcludeFormat: "--exclude '$folder/*' "
localBackupFolder: "/home/nc/backups/$serviceId/$manualOrAuto/$appName/$backupName"
restoreBackupCommand: "mkdir $folder && tar -C $folder -xvf $partName.tar.gz"
...
```

Pomocí konfigurační hodnoty **dirPath** definujeme systémové umístění dat pro *LXC kontejnery* (virtuální stroje). Hodnota **backupCommand** definuje příkaz do terminálu operačního systému pro vytvoření zálohy. Další konfigurační položka **backupExcludeFormat** udává formát pro předchozí příkaz, kterým se definují ignorované adresáře, které nebudou součástí výsledné zálohy. Pomocí **localBackupFolder** specifikujeme adresář pro finální umístění zálohy. Poslední hodnotou **restoreBackupCommand** definujeme příkaz pro obnovu dat z jedné části zálohy, počet částí závisí na konfiguraci konkrétní aplikace (nastavení záloh je popsáno v kapitole 3.4.1).

V ukázce jsou použity následující konfigurační proměnné:

- **$lxcDiskPool** - název diskového oddílu, který má nastavena *LXC* platforma
- **$name** - jméno části zálohy
- **$folder** - úplná cesta k adresáři, který má být zálohován
- **$excludedFolders** - proměnná obsahující formát definovaný v konfigurační hodnotě *backupExcludeFormat*
- **$manualOrAuto** - zálohy aplikací se dále dělí na manuální a automatické, tato hodnota obsahuje výčtovou hodnotu *"auto"* nebo *"manual"*, aby bylo možné odlišit, které zálohy uživatel prováděl manuálně a které byly vytvořeny automaticky

3.11.2 Obrazy disku (snapshots)

Vytváření obrazů celého virtuálního disku je možné pouze pro platformy, které pracují s virtualizací. Proto platforma *Direct* (3.3.1) tuto funkci nepodporuje. Modul platformy pro *LXC* virtualizaci (3.3.2) má vytváření obrazů disku implementované pomocí rozhraní z *PlatformAPI* (3.2.4), to umožňuje přistupovat k této funkcionalitě jednotným způsobem pro všechny moduly platforem. Pro vytvoření obrazů disku se využívá zabudovaná funkce *"lxc snapshot"* v virtualizačním software *LXC*, použitím parametru *–stateful* je možné zálohovat i stav operační paměti kontejneru společně se soubory na virtuálním disku.

```
Usage:
  lxc snapshot [<remote>:]<instance> [<snapshot name>] [flags]

Examples:
  lxc snapshot ul snap0
      Create a snapshot of "ul" called "snap0".

Flags:
      --no-expiry   Ignore any configured auto-expiry for the instance
      --reuse       If the snapshot name already exists, delete and create a new
 one
      --stateful    Whether or not to snapshot the instance's running state
```

```
Usage:
  lxc restore [<remote>:]<instance> <snapshot> [flags]

Examples:
  lxc snapshot ul snap0
      Create the snapshot.

  lxc restore ul snap0
      Restore the snapshot.

Flags:
      --stateful   Whether or not to restore the instance's running state from s
napshot (if available)
```

Obrázek 6: Nástroj "lxc snapshot"pro vytváření obrazů disků

Vytváření a obnova obrazů disku je plně automatizována a dostupná přes webové grafické rozhraní popsáno v sekci 6.7, tato ukázka příkazů do terminálu má pouze demonstrovat způsob, jakým lze s obrazy disku pracovat přímo v rozhraní LXC software.

4 Implementace web komponenty

Web komponenta slouží pro nasazování, ovládání a administraci služeb pomocí grafického uživatelského rozhraní.

4.1 Technologie

Web je postaven na technologii *Spring framework* ([9]) a jeho *MVC* (*Model-View-Controller*) systému. Komunikaci s databází zajišťuje framework *Hibernate* ([11]) a knihovna *Apache Commons DBCP* pak spravuje *connection pool* (sada předem připravených spojení s databází pro zrychlení procesu komunikace).

Pro prezentační vrstvu je použita šablona s názvem *AdminLTE* [12], je vyvíjena komunitně a má kompletně veřejný a otevřený zdrojový kód. Zároveň je také zdarma i pro komerční použití. Jako základ této šablony slouží HTML, CSS a JavaScript framework *Bootstrap 4* [**?**], který se v šabloně používá především pro rozložení a organizaci elementů na stránce. Nabízí také celou řadu připravených ovládacích a vzhledových prvků, tím značně zrychlí vývoj prezentační vrstvy webu. Použití šablony také sjednotí vzhled celého webu a je tak pro koncové uživatele přehlednější.

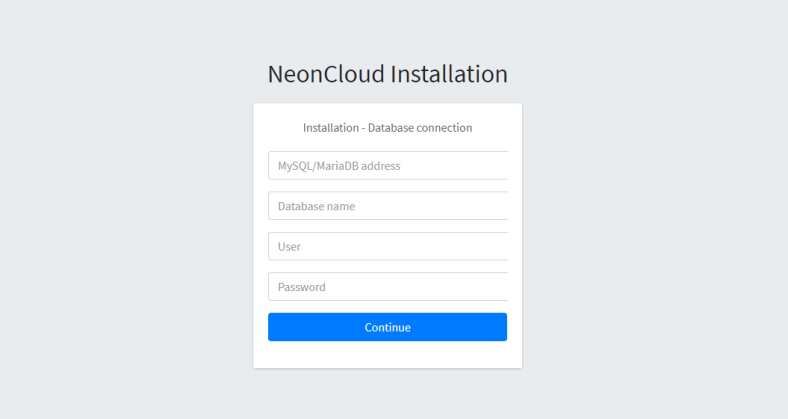
Pro poskládání finální podoby HTML je použit framework *Thymeleaf* [13], který dokáže složit finální HTML kód z několika částí, takže veškeré opakující se části jako hlavička, menu a patička stránky jsou definovány pouze na jednom místě a při jejich editaci se změny promítnou na všech stránkách. Pomocí *Thymeleaf* jsou také generovány veškeré tabulky, framework podporuje cykly a práci s proměnnými a lokalizací pomocí speciálních *HTML* atributů.

4.2 Lokalizace a jazyky

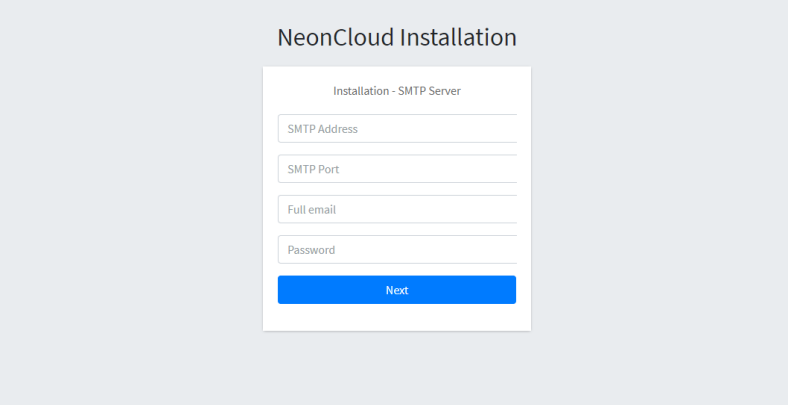
Web má zabudovanou podporu pro neomezený počet jazyků pro uživatelské rozhraní. Lokalizace lze dynamicky přidávat a editovat pomocí rozhraní webu (umožněno pouze uživatelům s administrátorskými oprávněními). Uživatelé si mohou konkrétní jazyk vybrat v horním menu dostupném na všech stránkách. Výchozí jazyk je angličtina, proto i ukázky z uživatelského rozhraní budou v této práci anglicky s českým popisem a překladem.

4.3 Instalace a první spuštění webové komponenty

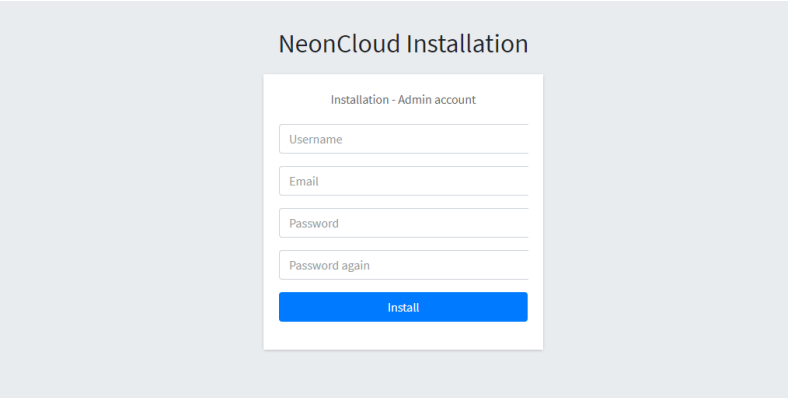
Sestavená webová komponenta je ve formátu *.WAR (Web Archive)* a není samostatně spustitelná, pro spuštění je vyžadováno nasazení na *JavaEE* (*Java Enterprise Edition*) web server (*Apache Tomcat, Glassfish* nebo jiný). Po nasazení není třeba nic konfigurovat, stačí otevřít adresu ve webovém prohlížeči a pokud web nenalezne konfigurační soubor (umístěný v adresáři *%USER_HOME%/neoncloud/config.yaml*), spustí se automaticky instalátor. Průběh instalace je zobrazen na následujících obrázcích 7, 8 a 9.



Obrázek 7: První krok instalace - údaje k databázi



Obrázek 8: Druhý krok - údaje k SMTP serveru (nevyžadováno)

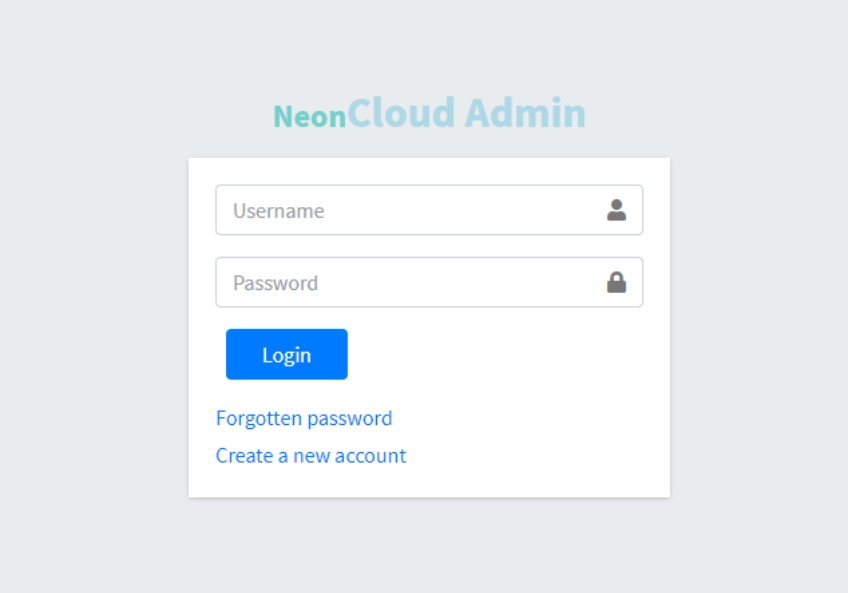


Obrázek 9: Třetí krok - vytvoření prvního účtu administrátora

Po instalaci dojde k vytvoření konfiguračního souboru (%USER_HOME%/neoncloud/config.yaml), datové struktury v databázi a naplnění některých tabulek výchozími hodnotami. Poté je možné se do systému s vytvořeným administrátorským účtem přihlásit.

4.4 Přihlašování a zabezpečení

Přihlašování a zabezpečení je na webu řešeno pomocí *Spring security*, což je oficiální rozšíření pro *Spring framework*.



Obrázek 10: Přihlášení do systému

Pokud jsme při instalaci přidali i SMTP server, pak je dostupná také funkce zapomenutého hesla a změny hesla na profilu uživatele s ověřením přes emailovou adresu. V případě zapomenutého hesla je uživatel vyzván k vyplnění své emailové adresy, pokud v systému účet s zadanou email adresou existuje, pak je na adresu zaslán odkaz URL s vygenerovaným kódem jako parametrem, pokud uživatel na daný odkaz přejde, zobrazí se mu formulář pro vyplnění nového hesla. Podobně při změně hesla, uživateli přijde na emailovou adresu email s potvrzovacím kódem. Uživatel zadá potvrzovací kód do formuláře, poté co vyplnil současné heslo a nové heslo, které si přeje nastavit.

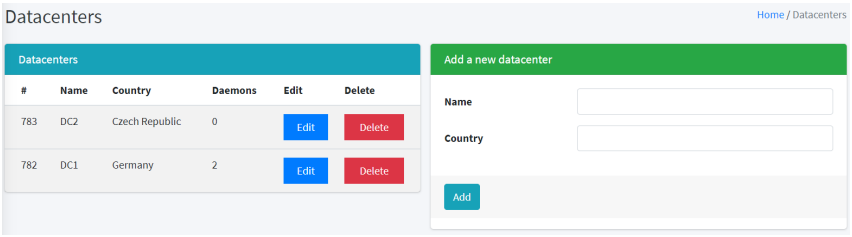
5 Administrace a konfigurace web komponenty

Před použitím systému, je třeba provést některá základní nastavení. PO úspěšné instalaci web komponenty je třeba vytvořit skupinu daemon instancí označovanou pojem "datacentrum", dále připojit nainstalované daemon instance, vytvořit hardware konfigurace a přidat aplikace, které jsme definovali na straně daemon komponent v kapitole 5.4. Posledním krokem je přidání plánu, který slouží jako předpis pro vytváření *IaaS* nebo *PaaS* služeb.

5.1 Datacentra

Prvním krokem po úspěšném nainstalování a přihlášení do webového ovládacího panelu je potřeba přidat **Datacentrum**, tímto pojmem je označena skupina nainstalovaných daemon instancí jejich hostitelské servery se nachází fyzicky ve stejné lokalitě, popřípadě ve stejné budově datacentra. Účel této funkce je usnadnit organizaci fyzických serverů a rozdělit je do skupin podle umístění. Rovněž je tím usnadněn přehled pro koncové uživatele nebo zákazníky, kteří si při objednávání mohou vybrat lokalitu, pokud je objednávaný plán služby dostupný ve více lokalitách (datacentrech).

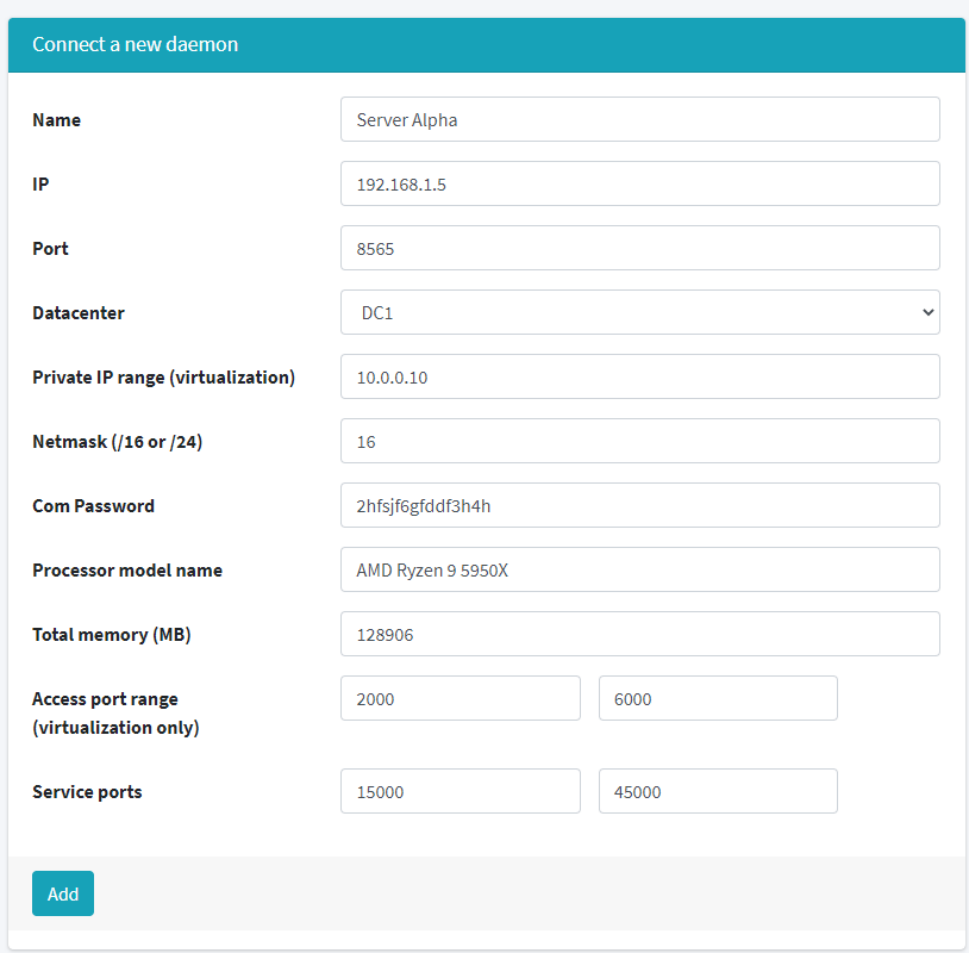
K vytvoření nového datacentra klikneme na položku *"Datacenters"* v hlavním menu, poté se zobrazí seznam existujících datacenter a formulář pro přidání nového datacentra. Entita datacentrum je velmi jednoduchá a obsahuje pouze 3 atributy, automaticky generované unikátní ID (Identifikátor), jméno a název lokality.



Obrázek 11: Přehled existujících datacenter a formulář pro přidání dalšího datacentera

5.2 Připojení daemon komponenty

Není možné vytvářet služby, dokud nepřipojíme alespoň jednu nainstalovanou instanci daemon komponenty, která odpovídá jednomu fyzickému hostitelskému serveru. Připojení nové daemon komponenty najdeme pod položkou *"Daemons"* v hlavním menu. Během připojování je nutné, aby komponenta byla spuštěna a TCP port *HTTPS* serveru (výchozí 8565) nebyl blokován žádným síťovým firewall software.



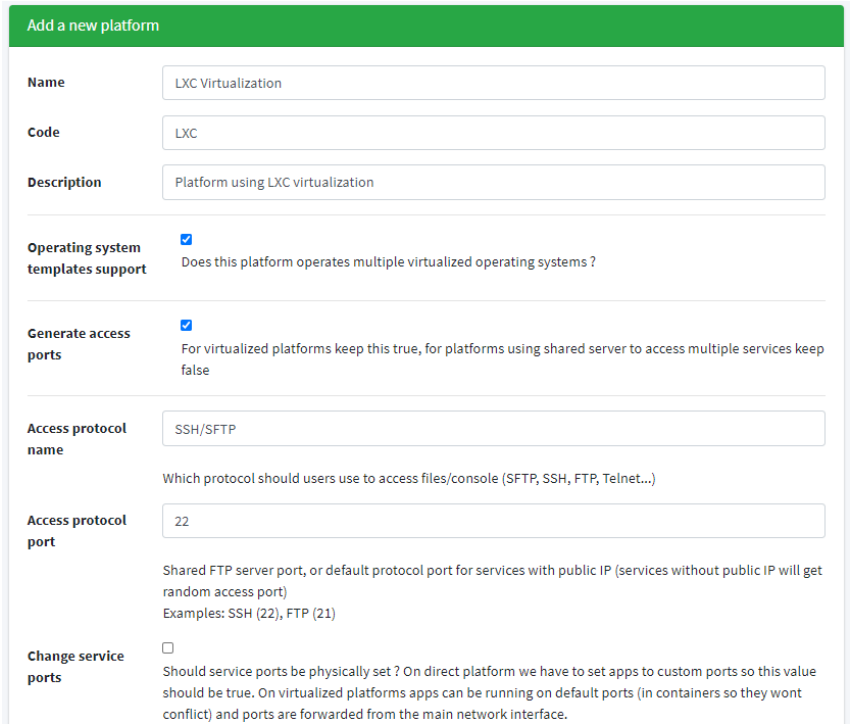
Obrázek 12: Rozhraní pro připojení hostitelského serveru - daemon komponenty

Jméno daemon instance specifikujeme položkou **Name**, jméno slouží pouze pro identifikaci jednotlivých instancí a je zobrazeno pouze administrátorům. Položkami **IP** a **Port** nastavíme IP adresu na které poslouchá *HTTPS* server na straně daemon komponenty, výchozí port je TCP/8565. Dále pomocí nabídky **Datacenter** vybereme datacentrum, do kterého chceme tento server umístit. Postup vytvoření nového datacentra je popsáno v předchozím bodě 5.1. **Private IP range** a **Netmask** specifikuje rozsah lokálních *IPv4* adres, které budou přiděleny virtuálním strojům pro lokální komunikaci, tento rozsah musí odpovídat konfiguraci virtualizačního software, který je na daném serveru nainstalován. **Processor model name** a **Total memory** jsou hodnoty ryze informativního charakteru, měly by obsahovat jméno použitého procesoru, respektive celkovou dostupnou operační paměť serveru. Hodnotou **Access port range** definujeme rozsah TCP/UDP portů pro zajištění přístupu k vytvořeným službám pomocí protokolů *SFTP, SSH* nebo *FTP* (záleží na implementaci v konkrétním platformním modulu). Vztahuje se pouze na virtuální stroje bez vlastní veřejné IP adresy, přemostění portů zajišťuje použitý modul platformy. Virtuální stroje s veřejnou IP adresou porty z tohoto rozsahu nepotřebují. Poslední hodnota **Service ports** taktéž slouží pro virtuální stroje pouze s lokální IP adresou, z

tohoto rozsahu se přiřazují TCP/UDP porty pro aplikace v rámci *PaaS* služeb. Virtuální stroje s veřejnou IP opět tento rozsah nevyužívají.

5.3 Přidání platformy

Do systému je nutné také přidat podporované platformní moduly (3.3) a dodatečné nastavení. To provedeme přes webové rozhraní výběrem položky hlavního menu *"Platforms"*. Po základní instalaci budou 2 moduly platforem již předinstalovány a přednastaveny: *LXC* a *Direct*.



Obrázek 13: Přidání platformy na straně webu

Parametry **Name** a **Description** slouží pro pojmenování respektive popis dané platformy. Hodnota **Code** je potřeba pro spárování webové konfigurace s konfigurací platformního modulu na straně daemon komponenty. **Operating system templates support** je boolovská hodnota, pokud nabude hodnoty pravda (políčko je zaškrtnuto), pak platforma podporuje více různých operačních systému, to znamená, že se jedná o platformu pracující s virtualizací. Další boolovská hodnota **Generate access ports** určuje, zda je potřeba pro službu alokovat TCP/UDP porty z rozsahu definovaném v nastavení konkrétní daemon instance (5.2) pro základní přístup pro uživatele, to se však týká pouze služeb, které budou ze sítě Internet dostupné pouze přes síťový most nebo *NAT*, služby s dedikovanou veřejnou IP adresou tuto možnost nevyužijí. Následující hodnota **Access protocol name**, má pouze informativní charakter, informuje uživatele o tom, jaký protokol mají použít pro připojení ke službě pro její správu, typicky zde bude protokol FTP, SFTP nebo SSH.

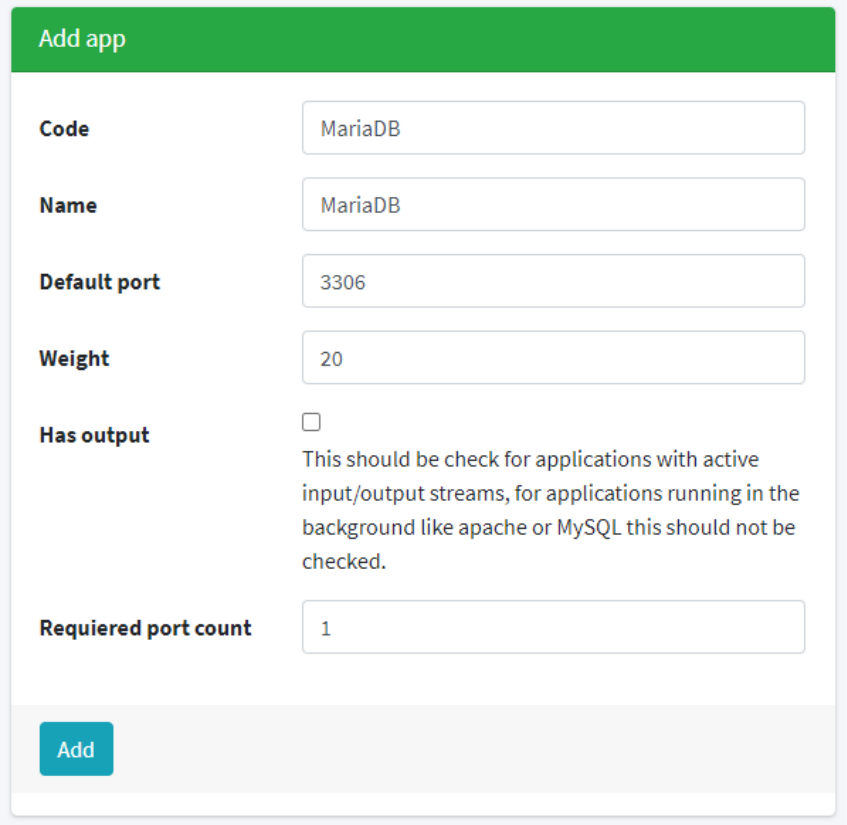
Access protocol port je důležité nastavení, v závislosti na využité platformě a na přiřazení veřejné nebo pouze lokální IP ke službě má vždy jiný význam:

- **Platforma nevyužívající virtualizaci** - Pokud se jedná o platformu, která nevyužívá virtualizaci, například platforma *Direct* (3.3.1) implementována v této práci, pak se bude jednat o nastavení portu ke sdílenému přístupovému serveru. To v praxi znamená, že se například na jedinou instanci *Apache2* web serveru nainstaluje několik zákaznických webových stránek a všichni zákazníci přistupují k souborům webu přes jeden sdílený FTP server, každý ale k jinému adresáři a adresáře ostatních uživatelů nejsou viditelné. Tato hodnota uvádí port tohoto sdíleného *FTP* nebo jiného přístupového serveru.
- **Virtualizovaná platforma a služba s veřejnou IP adresou** - Služba s vlastní veřejnou IP adresou má dostupné všechny TCP/UDP porty ze sítě Internet, pak tato hodnota reprezentuje výchozí port pro přístupový protokol. Služby s veřejnou IP na platformě *LXC* využívají protokol *SSH* na výchozím portu TCP/22 a proto bude mít toto nastavení v tomto případě také hodnotu 22.
- **Virtualizovaná platforma a služba bez veřejné IP adresy** - Služby, které mají pouze lokální IP adresy a veřejnou IP sdílí s dalšími službami, mají přidělen jeden port pro přístupový protokol z rozsahu nastaveného v předchozím kroku 5.2 při přidávání daemon instance. Platforma *LXC* používá síťový most pro směrování portů na virtuální stroj, kde běží *SSH* server na výchozím portu TCP/22 a proto hodnota tohoto pole bude také v tomto případě 22, tím platforma dostane informaci na jaký port má přidělený port z rozsahu nasměrovat pomocí síťového mostu.

5.4 Přidání aplikace

Dalším krokem po úspěšném připojení daemon instance je přidání aplikací a jejich spárování s aplikacemi nastavenými na straně daemon komponenty. V této ukázce opět použijeme pro demonstraci aplikaci *MariaDB*, stejně jako kapitole 3.4.1, kde je popsána konfigurace na straně daemon komponenty. Přidání nové aplikace najdeme v hlavním menu pod položkou *"Apps"*.

Položka **Code** udává unikátní identifikátor aplikace, musí se shodovat s kódem uvedeným v konfiguračním souboru na straně daemon komponenty (3.4.1). Následující položka **Name** již může být libovolné zobrazované jméno v rámci web komponenty. **Default port** udává výchozí TCP/UDP port aplikace. Pomocí **Weight** definujeme váhu aplikace, což jeden ze způsobů jak limitovat počet a typ aplikací, které může uživatel na službu s daným plánem nainstalovat, nastavení váhy pro plány je dále popsáno v následující kapitole 5.6. Polem **Has output** nastavíme, zda se u této aplikace bude zachytávat vstupní/výstupní proud, který bude možné na webu zobrazit nebo do něj poslat textový vstup. Tuto funkci využijeme u aplikací, které poskytují dodatečné ovládání přes příkazy do konzole. Poslední položkou **Required port count** nastavujeme počet TCP/UDP portů, které aplikace potřebuje. Pokud služba nemá veřejnou IP



Obrázek 14: Přidání aplikace na straně webu

adresu a běží za *NAT* nebo síťovým mostem (záleží na použité platformě a virtualizaci), toto číslo značí kolik portů daná aplikace potřebuje přesměrovat k tomu, aby byla dostupná přes síťové rozhraní. Služby s veřejnou IP adresou toto nastavení ignorují.

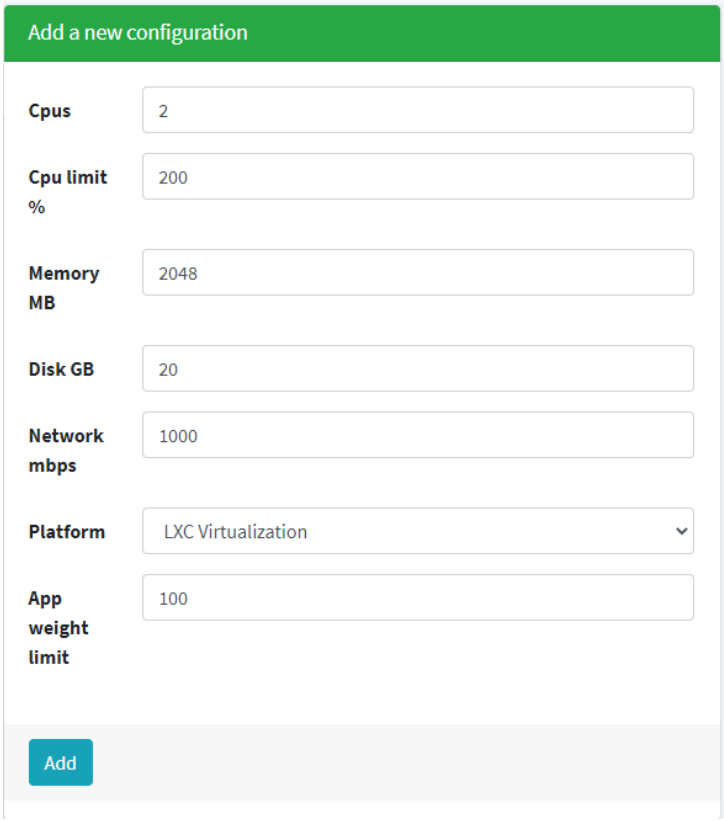
5.4.1 Přidání uživatelem definovaných proměnných

Po vytvoření nové aplikace, lze k aplikaci přidat uživatelem definované proměnné, které uživatel zadává před instalací tohoto typu aplikace. Proměnné jsou dostupné ve všech instalačních operacích definovaných v konfiguračních souborech daemon instance (použity například na ukázce v kapitole 3.4.1).

Každá proměnná má pevně definovaný datový typ, který je ověřen při zadávání, aby uživatel nemohl udělat chybu při zadávání a způsobit tak nefunkčnost instalačních skriptů. Aktuální podporované datové typy pro uživatelské proměnné jsou: *INT, DOUBLE, STRING, FLOAT, BOOLEAN*. Proměnné a jejich datový typ se specifikuje při vytváření proměnné pro aplikaci v grafickém rozhraní, které je dostupné po otevření detailů dané aplikace.

5.5 Přidání hardware konfigurace

Před přidáním plánu je potřeba nejdříve vytvořit hardware konfiguraci. Nastavení omezení hardware prostředků se nepřidává přímo do plánu, ale vytváří se samostatně jako entita konfigurace z důvodů znovupoužitelnosti u více různých plánů. Přidání nové konfigurace najdeme v hlavním menu pod položkou *"Configurations"*.

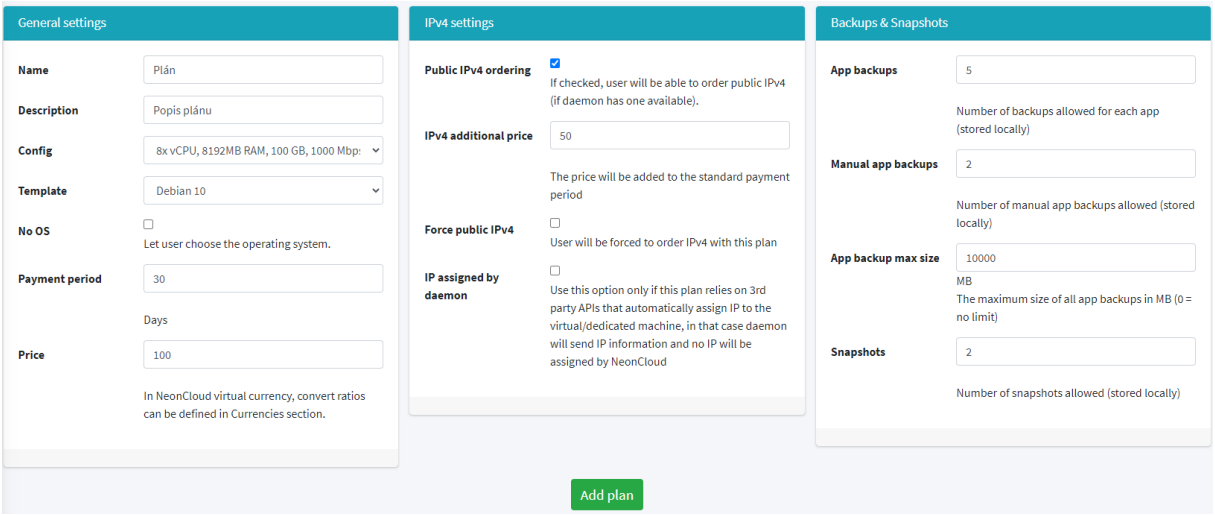


Obrázek 15: Přidání nové konfigurace

První hodnotou **Cpus** (zobrazenou na obrázku 15) specifikujeme počet virtuálních procesorů. Následující hodnota **Cpu limit %** nastavuje maximální využití procesoru v procentech, stejně jako v prostředí Linux 100% odpovídá využití celého jednoho jádra, 200% pak 100% využití 2 jader. **Memory MB** přiděluje operační paměť RAM (jednotka MB). **Disk GB** definuje velikost disku v jednotce GB. **Network mbps** limituje maximální přenosové rychlosti přes síťové rozhraní virtuálního stroje. Položka **Platform** nastaví platformu (3.3), pro kterou je tato konfigurace určena. Poslední položkou **App weight limit** definujeme maximální váhu všech instalovaných aplikací, hodnota bude použita pouze v případě, že se jedná o plán podporující služby typu *PaaS* a je možné na plán instalovat aplikace. Váhu aplikace specifikujeme během přidávání aplikací jak bylo popsáno v předchozí podkapitole 5.4.

5.6 Vytvoření plánu

Plán slouží jako předpis pro službu, podle kterého bude služba vytvořena. Plán určuje platformu, hardware konfiguraci, podporované aplikace a nastavení zálohování. Plán také obsahuje další parametry, které už se vztahují ke koncovým uživatelům jako je nastavení ceny a zúčtovacího období a nastavení zálohování. Nastavení plánů najdeme v hlavním menu pod položkou *"Prepared Plans"*.



Obrázek 16: Vytvoření nového plánu

Obrázek 16 zobrazuje formulář pro vytvoření nového plánu. Položky **Template** a **"No OS"** určují, zda se bude jednat o typ služby *IaaS* nebo *PaaS*, pokud zaškrtneme pole (hodnota pravda) **"No OS"**, bude se jednat o službu typu *IaaS* a uživatel bude moci po založení služby vybrat operační systém. V případě *PaaS* služeb si uživatel operační systém nevybírá a je při vytváření nainstalován automaticky podle hodnoty **Template**. Pro služby typu *PaaS* není možné vybrat operační systém, protože by bylo nutné nakonfigurovat instalační proces všech aplikací pro každý podporovaný operační systém zvlášť, instalace jednotlivých aplikací se i na různých distribucích operačního systému Linux může výrazně lišit. Pokud chceme podporovat například web server *Apache2* na distribuci *Debian* i *Fedora*, je možné vytvořit 2 různé plány a umožnit tak koncovému uživateli výběr mezi různými distribucemi i pro *PaaS* službu.

Po vytvoření plánu lze přidat seznam podporovaných aplikací, které uživatel může automaticky instalovat pomocí grafického uživatelského rozhraní. Grafické rozhraní pro nastavení seznamu podporovaných aplikací je dostupné po vytvoření plánu. Přidání aplikací je možné pouze u plánů, které mají staticky definovaný operační systém a jedná se tak o plán typu *PaaS*.

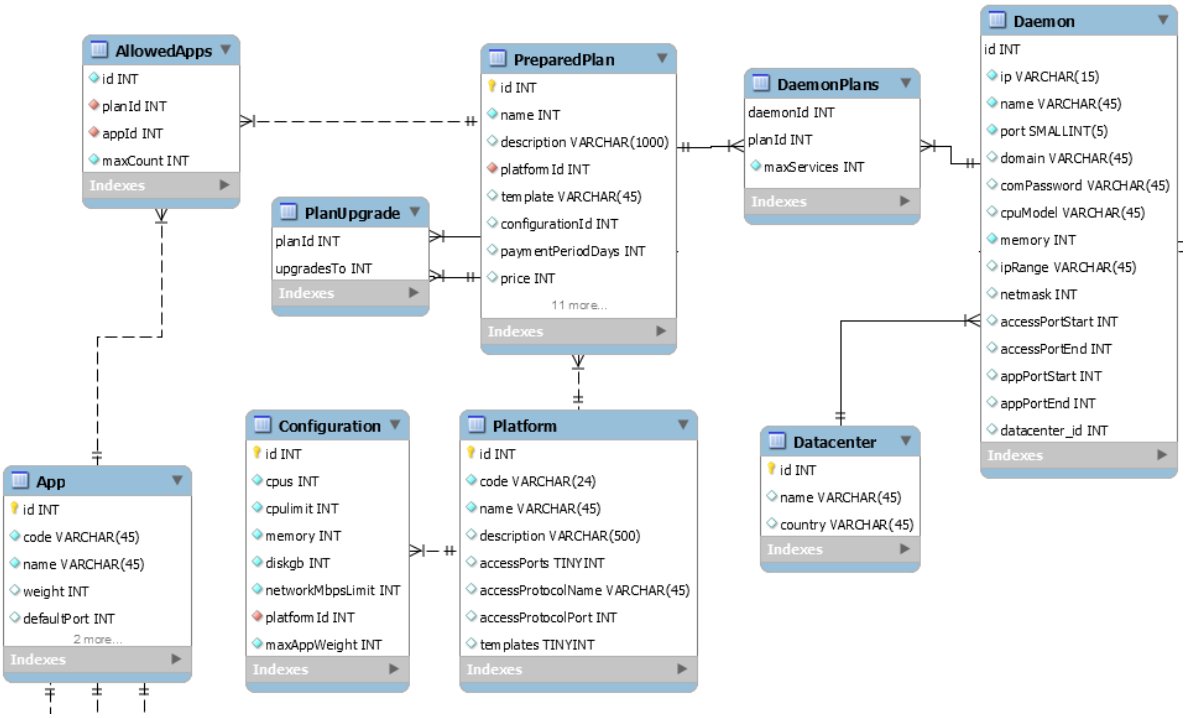
Pro každý vytvořený plán je možné také definovat plány, na které je možné plán vylepšit. Plány lze pouze povýšit na plán s vyšší hardware konfigurací, snižovat konfigurace není možné, protože při zmenšení velikosti virtuálního disku by mohlo dojít k poškození dat na virtuálním stroji.

5.6.1 Konfigurace podporovaných plánů

Po vytvoření plánu je třeba plán přiřadit k připojeným daemon instancím a určit kolik služeb s tímto plánem je možné na daný fyzický hostitelský server nasadit. Během přiřazení se provádí kontrola kontaktováním dané instance daemon komponenty a ověří se, zda skutečně je platforma (v případě *PaaS* plánu také všechny podporované aplikace) podporována a nakonfigurována, poté dojde ke spárování webové a lokální konfigurace plánu (případně i aplikací), tím je daná daemon instance připravena k nasazení služeb pomocí tohoto plánu.

5.6.2 Datový model

Datový model zobrazuje uložení entit Datacentra (*Datacenter*), Daemon instancí (*Daemon*), Konfigurace (*Configuration*), Platforem (*Platform*), Plánu (*PreparedPlan*) a Aplikací (*App*) v relační databázi. Model obsahuje pouze entity popsány v kapitolách 5.2 až 5.6. Pomocná entita *AllowedApps* obsahuje seznam povolených aplikací, které lze na daném plánu instalovat. Pomocná entita *DaemonPlans* obsahuje seznam podporovaných plánů pro danou daemon instanci.



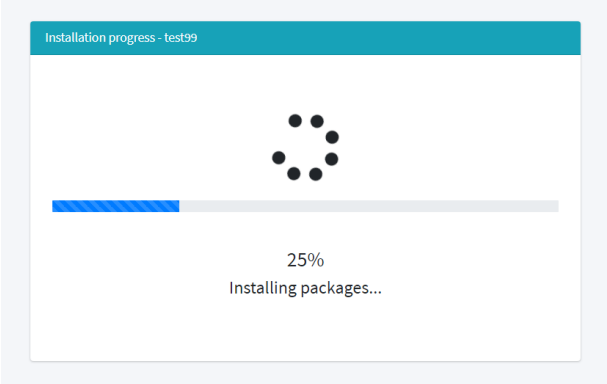
Obrázek 17: Část datového modelu zobrazující plány

6 Nasazování a správa cloud služeb

6.1 Vytvoření služby

Vytvoření služby je dostupné, jakmile je v systému nakonfigurován alespoň jeden plán a je připojena alespoň jedna daemon komponenta, která má nastavenou podporu pro minimálně jeden plán.

Pro vytvoření nové služby vybereme v hlavním menu položku *"+ Add Service"*, dále zvolíme instanci daemon komponenty, na kterou chceme službu nasadit. Dále službu vhodně pojmenujeme a vybereme plán, podle kterého bude vytvořena. V případě služby typu *PaaS*, která má pevně nastavenou šablonu operačního systému, začne ihned po vytvoření automatická instalace virtuálního stroje, uživatel vidí přibližný postup instalace živě v prostředí webového grafického uživatelského rozhraní (ukázka na obrázku 18).



Obrázek 18: Instalace virtuálního operačního systému v reálném čase

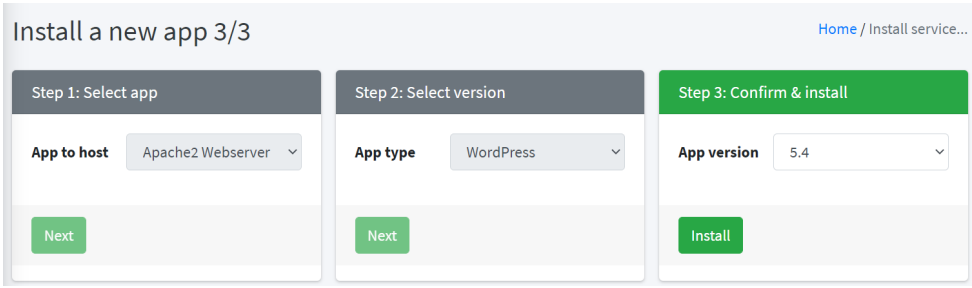
V případě, že se jedná o *IaaS* plán, je uživatel vyzván k výběru operačního systému, který chce instalovat. V momentě psaní této práce, systém aktuálně má předkonfigurovanou podporu pro následující systémy: *Debian 10, Debian 9, Devuan 3, CentOS 8, Fedora 33*. Přidání dalších distribucí je prováděno na straně jednotlivých instancí daemon komponent, nastavení šablony pro operační systém je pouze záležitost konfigurace a není potřeba nijak zasahovat do zdrojových kódu některé z komponent. Konfigurace operačních systémů byla popsána v kapitole 3.9.1.

6.2 Instalace aplikací

Pokud má plán povoleno instalovat aplikace, respektive jedná se o plán typu *PaaS*, může uživatel po založení služby instalovat vybrané aplikace (pokud má ke službě příslušná oprávnění).

Následující obrázek 19 zobrazuje nasazení aplikace *Apache2* s použitím šablony *Wordpress* ve verzi *5.4*. Pomocí tohoto grafického rozhraní může uživatel nasadit ve třech krocích jakoukoliv podporovanou aplikaci. Pokud má aplikace nastaveny uživatelem definované proměnné, pak je před samotnou instalací uživatel dotázán k vyplnění všech uživatelem definovaných proměnných, tyto proměnné jsou pak dostupné pro všechny instalační operace nakonfigurovány na straně

daemon komponenty (například v ukázce 3.4.1 jsou proměnné použity pro počáteční nastavení MariaDB/MySQL serveru).

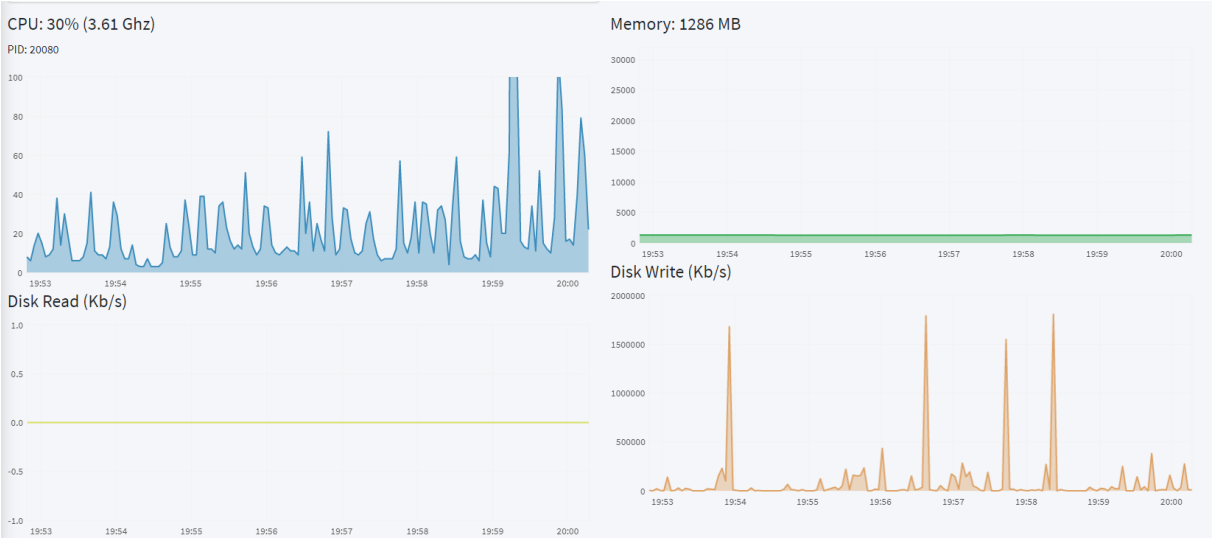


Obrázek 19: Ukázka instalace nové aplikace

Průběh instalace aplikace je taktéž možné sledovat živě se zobrazením přibližného postupu instalace v procentech podobně jako při instalaci samotného operačního systému, jak bylo ukázáno v kapitole 6.2.

6.3 Monitorování aplikací

Pro každou aktivní aplikaci běžící na popředí, jako je například *TeamSpeak 3 Server*, je možné v grafickém rozhraní web komponenty zobrazit v reálném čase grafy vytížení jednotlivých hardware prostředků. Systém monitoruje vytížení procesoru v procentech, alokovanou operační paměť a aktivitu disku (čtení a zápis).



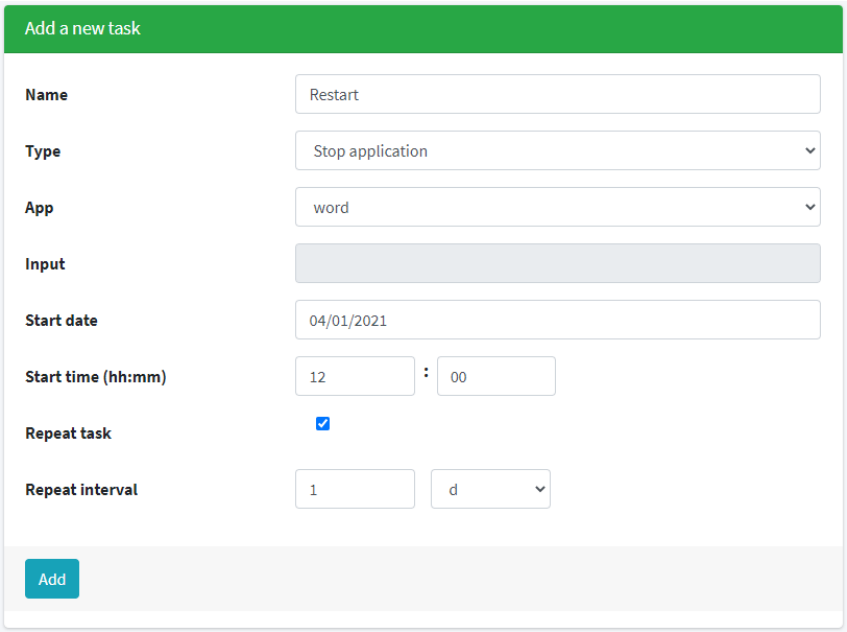
Obrázek 20: Monitorování aplikací v reálném čase

Pro vykreslování grafů je použita knihovna *Jquery Flot* [16]. Periodická aktualizace grafů je řešena pomocí *asynchronního Javascriptu (Ajax)*, který posílá HTTP/S požadavky na API web komponenty. Samotné monitorování a ukládání hodnot zajišťuje modul **Monitor** (3.2.6)

na straně daemon komponenty na základě *PID* (*Process ID*), hodnota *PID* je této komponentě předána z modulu **Executor** (3.2.5) během startování nainstalované aplikace, která běží na popředí.

6.4 Naplánované úlohy

Web nabízí grafické rozhraní pro plánování úloh. Úlohy se mohou vztahovat k službě nebo k nainstalované aplikaci v případě služby typu *PaaS*. O vykonávání a časování úloh se stará modul 3.2.8 na straně daemon instance.

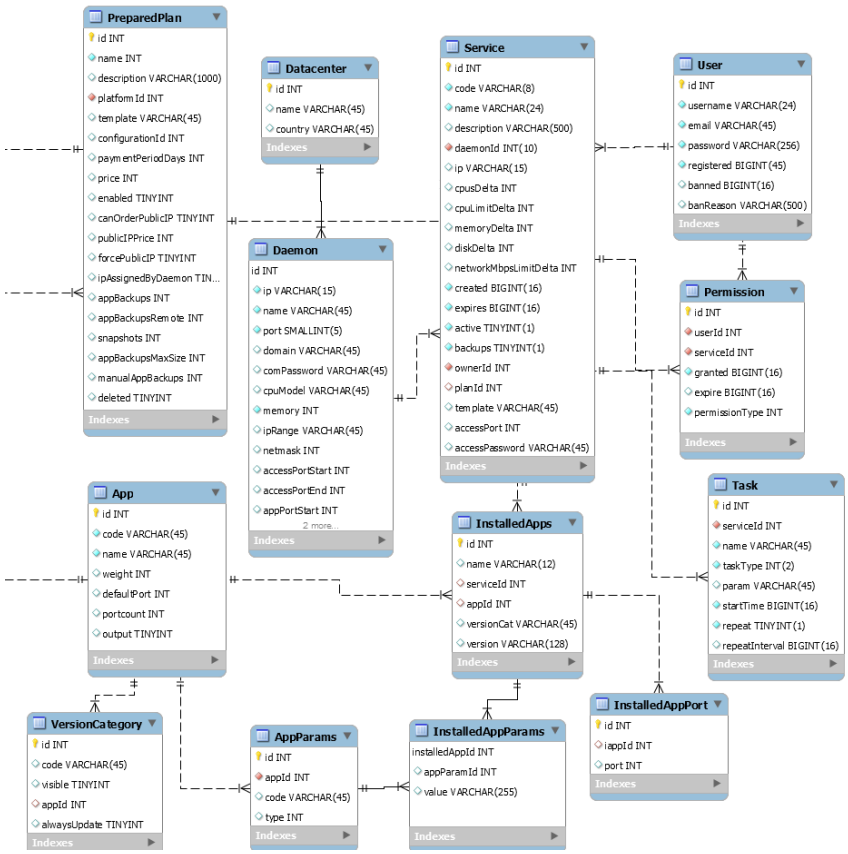


Obrázek 21: Rozhraní pro vytvoření nové úlohy

Polem **Name** může uživatel úlohu pojmenovat, následující rozbalovací seznam **Type** slouží pro výběr typu úlohy. Typy úloh byly popsány v kapitole 3.2.8. Následující rozbalovací seznam označen jako **App** slouží pro výběr nainstalované aplikace, v případě, že se jedná o úlohu, která se vztahuje k aplikaci. Následující pole **Input** je neaktivní, a je aktivováno pouze v případě, že se jedná o úlohu, která zasílá vstup do vstupního proudu aktivní aplikace. Pomocí pole **Start date** definujeme datum prvního spuštění úlohy, ten dále specifikujeme zadáním času přes pole **Start time**. Zaškrtnutím pole **Repeat task** se aktivuje poslední položka **Repeat interval**, pomocí které nastavíme čas opakování úlohy.

6.5 Datový model služby

Část datového modelu, která zobrazuje strukturu entit a vazby mezi nimi. Je zobrazena pouze část entit, které se přímo vztahují k entitě *Service* (služba). Entita *PreparedPlan* uchovává informace o plánech (5.6) pro služby. Služba může mít pouze jeden plán. Entity *InstalledApps, AppParams, InstalledAppParams a InstalledAppPort* se vztahují k *PaaS* službám a nainstalovaným aplikacím v rámci těchto služeb. Tabulka *Task* uchovává entity naplánovaných úloh. Služba může mít jednoho uživatele (tabulka *User*), který je majitelem služby a několik uživatelů, může mít ke službě přiděleno oprávnění pomocí entit v tabulce *Permission*.

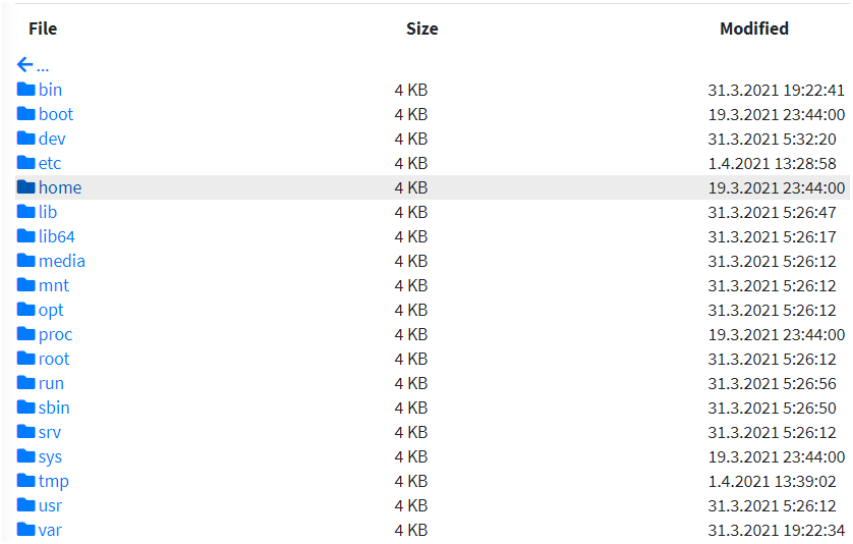


Obrázek 22: Datový model služby

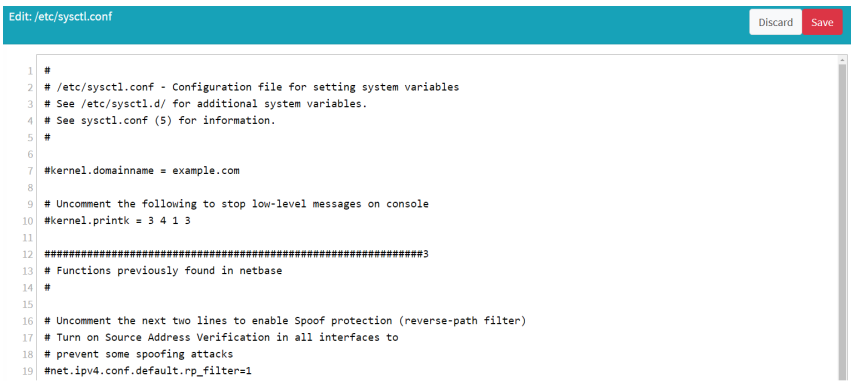
Jedná se pouze o část datového modelu, který se vztahuje k entitám popsaných postupně v kapitole 6. Kompletní datový model se nachází v příloze A.

6.6 Správce souborů

Web rozhraní nabízí možnost procházet, zobrazit, vytvářet nebo editovat soubory. Funkce se může hodit pro rychlou kontrolu log souborů v případě, že administrátor služby momentálně nemá k dipozici SSH nebo SFTP klienta.



Obrázek 23: Správce souborů - procházení adresářové struktury



Obrázek 24: Správce souborů - editor

Správce souborů je implementován v jednotlivých platform modulech na straně daemon komponenty využívající API z modulu **PlatformAPI** 3.2.4, tím je zajištěno, že se se správcem souborů pracuje na všech platformách stejně. Obě implementované platformy v rámci této práce (*Direct* a *LXC*) využívají pro editaci souborů znakovou sadu *UTF-8*.

6.7 Zálohování

Systém zálohování je možné plně ovládat pomocí web rozhraní u služeb, které mají zálohování povoleno, to záleží na nastavení plánů, podle kterého je služba vytvořena (5.6). Zálohy jsou rozděleny do 2 kategorií:

- **Zálohy aplikací** zálohují pouze data pro konkrétní nainstalovanou aplikaci na PaaS plánech podle nastavení záloh v konfiguračním souboru na straně daemon instance (popsáno v kapitole 3.4.1). Zálohy aplikací se dále dělí na:
  - **Automatické zálohy** jsou vytvářeny pomocí naplánované úlohy (6.4). Pokud je přesažen limit záloh nebo limit celkové velikosti záloh, dojde ke smazání nejstarší automatické zálohy.
  - **Manuální zálohy** mohou být vytvořeny manuálně uživatelem. Zálohy mohou být opět smazány pouze uživatelem a nikdy nedojde k jejich automatickému smazání.

  Aplikace může být automaticky vrácena do stavu v době manuální či automatické zálohy pomocí funkce *Restore* (implementace popsána v kapitole 3.11.1). Funkce je fyzicky implementována v modulu platformy, který služba využívá (3.2.4).

- **Obrazy virtuálních disků** jsou dostupné pouze na platformě, která využívá virtualizaci a na plánech, které mají vytváření obrazů (*snapshot*) povoleno (5.6). Vytváření obrazů disků je časově i prostorově velmi náročné, proto nelze vytvářet obrazy pomocí automatických naplánovaných úloh. Obraz disku může vytvořit uživatel pouze manuálně, například před tím, než se pokusí provést změny v důležitém nastavení systému a chce mít jistotu, že v případě selhání může celý virtuální stroj obnovit do stavu před provedením změn. Implementační detaily jsou popsány v kapitole 3.11.2.

Zálohy jsou jediná entita v systému, která není uložena v databázi. Informace o dostupných zálohách se získává pouze z struktury souborového systému, kde jsou zálohy uloženy. Protože se předpokládá, že zálohované soubory budou v podobě jednoho, případně několika archivních souborech (záleží na konfiguraci aplikace, popsáno v kapitole 3.4.1), informace jako datum vytvoření zálohy a velikost lze zjistit časově i výkonově nenáročnou operací. Podoba záloh záleží na implementaci a konfiguraci konkrétního modulu platformy, nicméně moduly *Direct* (3.3.1) a *LXC* (3.3.2), implementovány a předem nakonfigurovány v rámci praktické části této práce ukládají zálohy v archivním formátu *tar* s kompresí *Gzip* (*tar.gz*).

7 Další funkce webového rozhraní

Funkce popisované v této kapitole jsou soustředěny na koncové uživatele grafického uživatelského rozhraní, týkají se především lokalizace, plateb a systému podpory. Tyto funkce nebyly součástí zadání této diplomové práce, nicméně pro reálné použití a nasazení celého systému do praxe jsou tyto funkce nezbytné.

7.1 Překlady a lokalizace

Systém je připraven na překlad HTML *Thymeleaf* [13] šablon. Překlady jsou založeny na funkcionalitě *Spring* modulu *LocaleResolver* z balíku *org.springframework.web.servlet*. O samotný překlad jazykových kódu se stará třída implementující abstraktní třídu *org.springframework.context.support.AbstractMessageSource*.

Změna jazyka se u klienta řeší pomocí zachycení parametru *"lang"* z URL adresy. Tento parametr je zachycen interceptorem *LocaleChangeInterceptor* z balíku *org.springframework.web.servlet.i18n* a jazyk klienta je uložen k ostatním datům sezení.

Překlady lze kdykoliv editovat přímo pomocí web rozhraní, změny se projeví okamžitě. K sekci z překlady se dá dostat přes hlavní menu pomocí položky *Translations*. Překlady jsou uloženy v databázi pro snadnější správu distribuovaného nasazení web komponenty.

7.2 Měny

Systém podporuje neomezené množství měn, jednotlivě se dají nakonfigurovat pomocí grafického rozhraní. U vytváření plánů v kapitole 5.6 jsme specifikovali cenu plánu, ale nikoliv měnu. Specifikace ceny u plánu totiž využívá "virtuální měnu", která se automaticky přepočítá na měnu, kterou má zvolenou koncový zákazník. V případě změny kurzu, tak není nutné manuálně měnit ceny ve všech podporovaných měnách pro každý plán, ale stačí pouze změnit kurz měny a ceny budou automaticky přepočítány.

Podle země uživatele se také může připočíst k finální ceně DPH (Daň z přidané hodnoty). Systém umožňuje specifikovat pravidla, kdy se k ceně DPH bude připočítávat a kdy ne. Toto nastavení může vycházet například podle země, ze které koncový uživatel pochází a zda ma na svém profilu vyplněno DIČ (Daňové identifikační číslo, anglicky označováno jako *VAT ID*). Toto nastavení je důležité pro koncové zákazníky, kteří jsou plátci DPH, aby nedošlo ke dvojitému zdanění.

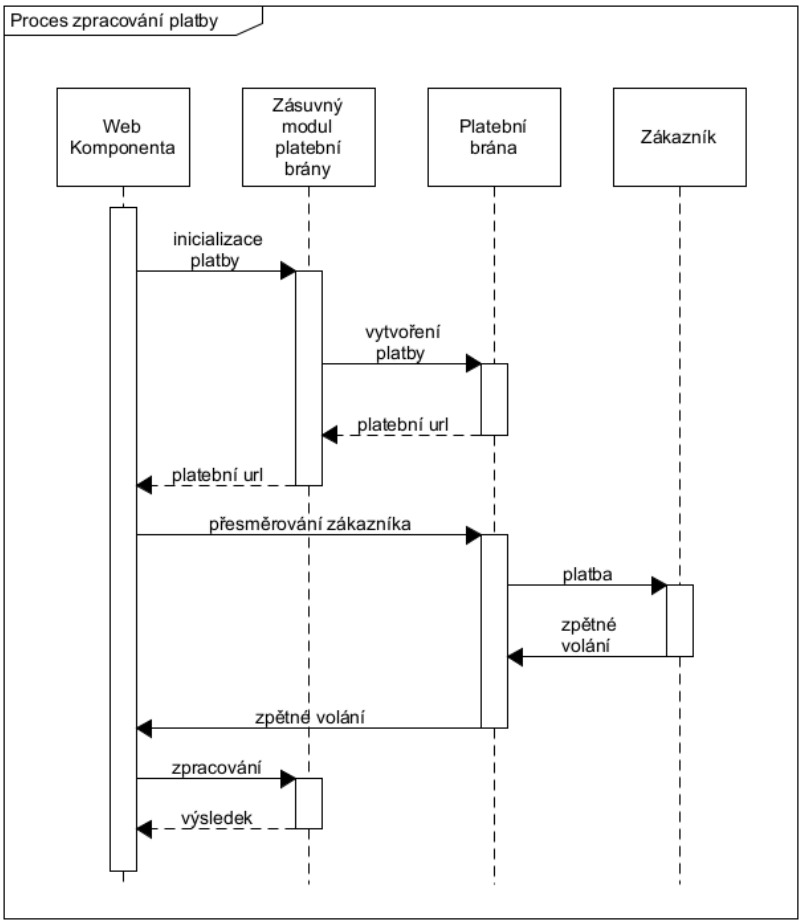
7.3 Mezinárodní validace daňového identifikačního čísla

Systém obsahuje implementaci pro automatické ověření *VAT ID* proti Evropskému VAT validátoru *VIES*: *https: //ec.europa.eu/taxation_customs/vies/*. Pro mezinárodní validaci je implementováno komerční API: *https://cloudmersive.com/VAT-number-validation-API*, které je však

zdarma do 800 požadavků měsíčně (informace k datu 10.4.2021), případně je možné zakoupit klíč s větším limitem a vložit ho do systému.

7.4 Systém zásuvných modulů pro platební brány

Protože dnes existuje mnoho způsobů, jak platit online, je vhodné řešit podporu dynamicky, tedy načítáním zásuvných modulů. Každý zásuvný modul reprezentuje jednu online platební bránu. Koncový zákazník si poté vybere způsob platby a podle toho dojde k výběru příslušného načteného zásuvného modulu a inicializaci platby. Inicializace platby spočívá u většiny online platebních bran zavoláním jejich *API* (typicky *REST API*) s detaily o platbě, jako částka, měna a údaje o zákazníkovi. Odpověď typicky obsahuje URL nebo HTML kód, pomocí kterého je uživatel přesměrován na web stránky provozovatele platební brány, kde dojde k platbě. Web provozovatele je oddělený, takže například údaje o kreditní kartě se k obchodníkovi, který tuto platební bránu provozuje, nedostanou a platby jsou tak pro zákazníky bezpečné.



Obrázek 25: Sekvenční diagram zpracování plateb

Sekvenční diagram na obrázku 25 demonstruje průběh platby pomocí systému zásuvných modulů, které volají *API* konkrétních platebních bran. Po vytvoření platby je zákazník přesměrován na platební bránu, kde provede transakci. Poté je přesměrován zpět a systém vygeneruje fakturu. Dokončení objednávky aktuálně (13.04.2021) ještě není implementováno.

7.5 Zabudovaný systém podpory

Systém podpory slouží pro komunikaci mezi koncovými uživateli a provozovatelem. Systém umožňuje vytvořit zprávu, kterou poté v systému uvidí pouze oprávnění lidé, tedy administrátor a člen týmu zákaznické podpory, podle tabulky rolí: 2.4. Na zprávu poté oprávněná osoba zašle odpověď. Zprávy mohou mít statusy:

- **Otevřeno** - Nová zpráva koncového uživatele
- **Zodpovězeno** - Uživatel obdržel odpověď od zákaznické podpory
- **Uzavřeno** - Zpráva je vyřešena a nelze na ní dále odpovídat

Pro rychlejší řešení problémů, je také možné definovat šablony odpovědí, ze kterých může člen zákaznické podpory automaticky vložit odpověď na často kladené dotazy a nemusí jí tak pokaždé psát ručně. Systém šablon odpovědí podporuje vícejazyčnost a šablonu odpovědi lze definovat pro všechny podporované jazyky.

7.6 Varovný systém

Varovný systém je stále ve vývoji (k datu 16.4.2021), momentálně umí zaznamenat pouze 2 typy problémů: výpadek služby a výpadek daemon komponenty. Po detekování problému je vytvořeno varovné oznámení, které lze vidět na hlavní straně web rozhraní hned po přihlášení. Výpadek služby vidí majitel služby, zatímco systémový problém, tedy výpadek celé daemon komponenty vidí pouze administrátor. V budoucnu by měl být systém rozšířen o další typy varování a také by mělo být uživateli umožněno zvolit si, které typy oznámení mají být zaslány také na emailovou adresu uživatele.

8 Testované aplikace pro PaaS

Díky podpoře systémových a uživatelem definovaných proměnných (3.5) a jejich integraci do instalačních skriptů aplikace (3.4.2) je možné přidat podporu pro nové aplikace pouze pomocí konfigurace a není tak třeba zasahovat do zdrojových kódů některé z komponent systému. V rámci této práce byly otestovány následující populární aplikace provozované na serverech.

8.1 Apache 2

Instalace web serveru *Apache2* je plně automatizovaná pomocí operací pro daný operační systém. Díky systému šablon je zajištěno, aby uživatel nainstaloval například *WordPress* nebo *MediaWiki* z připravených souborů šablony.

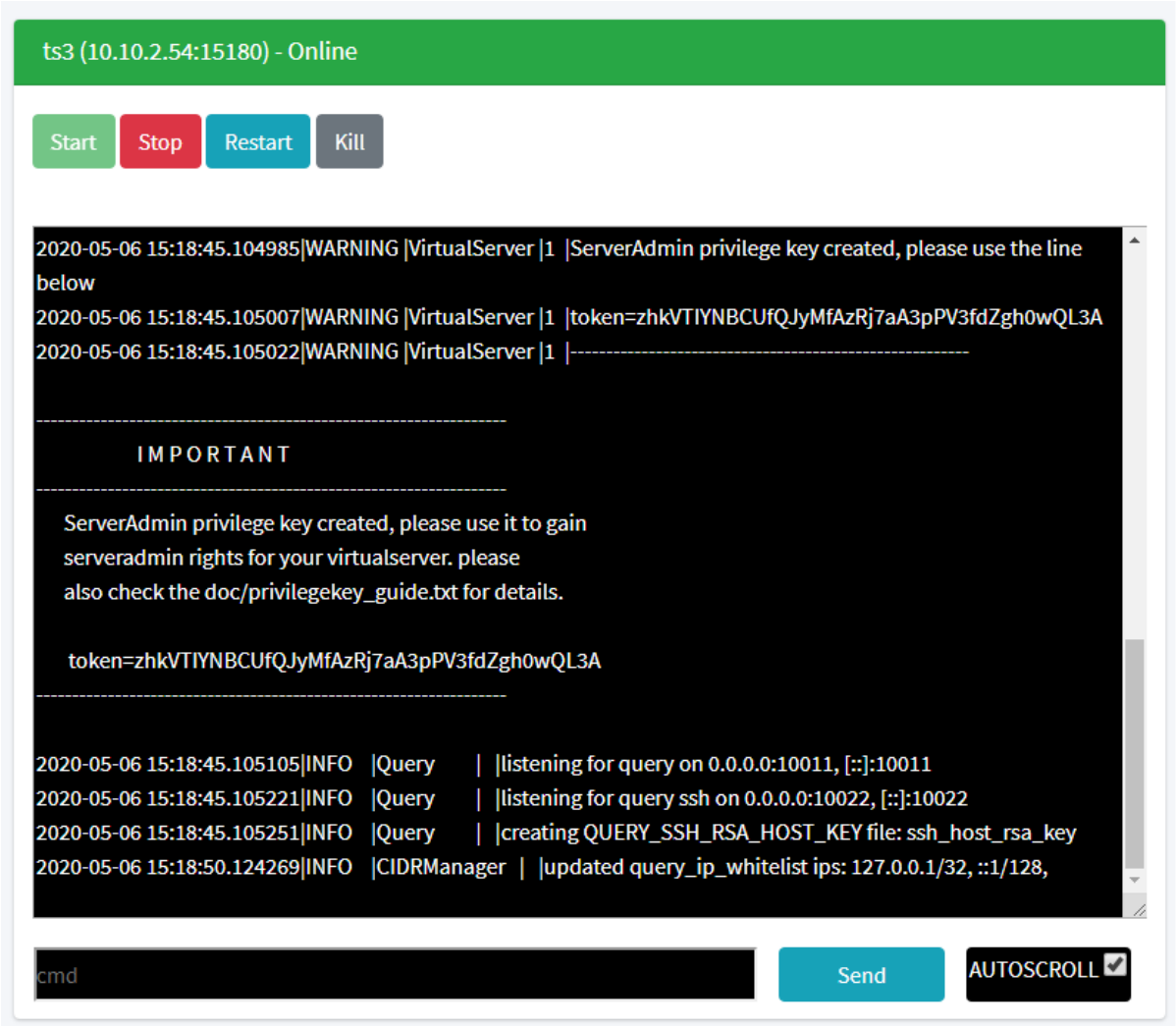
Pokud má služba přidělenou veřejnou IP adresu, pak bude *Apache2* web server dostupný přímo na výchozím portu (TCP:80), v opačném případě systém vybere jeden z volných portů a překonfiguruje port v konfiguračním souboru *Apache2* web serveru. V případě LXC platformy bude také vytvořen síťový most a port bude přesměrován na síťové rozhraní virtuálního stroje.

8.2 MariaDB / MySQL

MariaDB/MySQL je další testovaná aplikace v systému. Instalační proces je stejně jako u *Apache2* automatizován pomocí konfigurovatelných operací na daném operačním systému (3.4.1). Při instalaci *MariaDB* se využijí uživatelské proměnné pro definici jména a hesla pro nově nainstalovanou databázi. Tyto proměnné je možné použít na straně daemon komponenty v konfiguračních souborech pro definici příkazů pro vytvoření nové databáze a nastavení hesla.

8.3 TeamSpeak 3 Server

Další otestovanou aplikací v systému byl *TeamSpeak 3 server* pro známý komunikační program *TeamSpeak 3*, který umožňuje hlasovou komunikaci po síti. Zde je důležitá podpora zachytávání konzolového výstupu a předávání výstupu do "živé"konzole na webu, protože při prvním spuštění *TeamSpeak 3 server* vypíše na svůj standardní výstupní proud speciální klíč pro získání administrátorských oprávnění na *TeamSpeak 3 serveru*. Z bezpečnostních důvodů token není uložen v logu a tak zachycení výstupu je jediná možnost pro koncového uživatele, jak tento speciální klíč získat.



Obrázek 26: "Živá"konzole
Umožní sledovat konzolový (textový) výstup aplikace v reálném čase. Pokud daná aplikace podporuje vstupní příkazy, pak je možné také do konzole příkaz zapsat. Automatická aktualizace konzolového výstupu je zajištěna pomocí asynchronního JavaScriptu.

9 Závěr

Cíle této práce se podařilo naplnit a byl vytvořen funkční systém pro správu *PaaS* a *IaaS* cloud služeb. Implementace je modulární a tak do budoucna připravena na další rozšíření systému o nové funkce a možnosti. Díky modularitě a použití *OSGi* technologie na straně daemon komponenty lze nasadit důležitou aktualizaci bez přerušení služeb pro koncové uživatele.

V systému bylo otestováno několik aplikací, včetně skupiny často označované jako *LAMP (Linux, Apache, MariaDB, PHP)*, všechny tyto aplikace jsou podporovány pouze na bázi konfigurace systému, není nutné vytvářet další modul do nebo měnit nějaké části zdrojových kódů, jejich instalace a nastartování je pro koncového uživatele záležitost téměř jednoho kliknutí ve webovém ovládacím panelu.

9.1 Plánované funkce

I když je systém plně funkční a dá se již nasadit jako hotové řešení, stále jsou funkce, které se do této práce nevešly a funkčně by systému pomohly. Mezi ně patří migrace, která by umožnila kdykoliv přesunout službu z jednoho fyzického hostitelského serveru na jiný a pro koncového uživatele by se nic nezměnilo.

Aktuálně systém podporuje pouze *IPv4* veřejné adresy. V prostředí datacenter je stále dost volných *IPv4* adres, nicméně do budoucna je přidání podpory *IPv6* klíčové.

Podpora hromadných operací by dále usnadnila práci administrátorům, kteří spravují více služeb, hromadné operace by umožnily hromadný start nebo restart několika služeb najednou, či hromadné naplánování záloh.

Na správu více služeb najednou navazuje také škálování. Systém by v budoucnu měl podporovat automatické škálování aplikací, které by probíhalo automaticky podle zadaných podmínek. V případě web serveru *Apache2* by jako podmínka mohl být počet HTTP/S požadavků za určitý časový interval, v případě přesažení limitu by došlo k vytvoření další instance *Apache2* web serveru.

Poslední plánovaná funkce by měla rozšířit stávající zálohovací systém o zálohy na externí server. Přestože většina serverů používá diskové pole *RAID* a je chráněna proti chybě jednoho z disků, v případě živelné katastrofy o data ale můžeme přijít.

Literatura

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Gang of Four): Návrh programů pomocí vzorů. Grada. Praha 2003. ISBN 8024703025

[2] Craig Walls (Manning Publications), Spring in action (5. vydání). Shelter Island, NY 2019. ISBN 978-1617294945

[3] DARWIN, Ian F. Java cookbook. 2nd ed. Sebastopol, CA: O'Reilly, c2004, xxiv, 829 p. ISBN 05-960-0701-9.

[4] OSGi Framework [online] [cit. 4.4.2021]. Dostupné z: https://www.osgi.org/resources/what-is-osgi/

[5] Historie Amazon EC2 [online] [cit. 19.3.2021].

Dostupné z: https://aws.amazon.com/blogs/aws/ec2-instance-history/

[6] Hetzner Cloud [online] [cit. 19.3.2021]. Dostupné z: https://www.hetzner.com/news/viel-cloud-fuer-wenig-geld-hetzner-online-definiert-cloud-neu/

[7] Požár v datacentru společnosti OVHcloud [online] [cit. 19.3.2021]. Dostupné z: https://www.ovh.ie/news/press/cpl1786.strasbourg-datacentre-latest-information

[8] Vyjádření ředitele společnosti OVHcloud [online] [19.3.2021].

Dostupné z: https://www.youtube.com/watch?v=YGhkM_-e9sY

[9] Spring Framework [online] [cit. 14.1.2021]. Dostupné z: Spring MVC (Model View Controller). https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html

[10] Boostrap 4 HTML framework [online] [cit. 16.1.2021].

Dostupné z: https://getbootstrap.com/

[11] Hibernate ORM [online] [cit. 21.1.2021]. Dostupné z: https://hibernate.org/

[12] AdminLTE šablona [online] [cit. 24.1.2021]. Dostupné z: https://adminlte.io/

[13] Thymeleaf, šablonovací systém pro JavaEE [online] [cit. 3.2.2021]. Dostupné z: https://www.thymeleaf.org/

[14] OSHI - Multiplatformní hardware monitor API [online] [cit. 4.1.2021]. Dostupné z: https://github.com/oshi/oshi

[15] Java 9 nové process API [online] [cit. 2.3.2021]. Dostupné z: https://www.baeldung.com/java-9-process-api

[16] Jquery Flot knihovna. [online] [cit. 31.3.2021]. Dostupné z: https://www.flotcharts.org/

Přílohy

A Datový model

Příloha v IS EDISON. Soubor *data-model.png* obsahuje datový model ve formátu *PNG*.

B Zdrojové kódy aplikace

Příloha v IS EDISON. Adresář *Zdrojové kódy* obsahuje veškeré zdrojové kódy aplikace web a daemon komponenty.

C Návod na instalaci

Příloha v IS EDISON. Soubor *instalace.html* obsahuje textový a video návod na instalaci daemon a web komponenty, jejich spojení, nastavení a vytvoření první služby.

D Sestavená daemon komponenta

Příloha v IS EDISON. Adresář *Daemon* obsahuje poslední sestavenou verzi daemon komponenty připravenou ke spuštění.