Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Raspberry Pi jako IT infrastruktura pro malé a střední podniky

Bc. Vítězslav Košina

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Vítězslav Košina

Systémové inženýrství a informatika
Informatika

Název práce

Raspberry PI jako IT infrastruktura pro malé a střední podniky

Název anglicky

Raspberry Pi as IT infrastructure from Small and medium-sized enterprises

---

Cíle práce

Cílem práce je vytvořit konfiguraci clusteru z několika Raspberry PI, které díky enterprise distribuci Linuxu poskytnou zaměstnancům i klientům zabezpečené IT služby pro intranet, extranet i internet, jak v rámci vnitřní sítě, tak i vzdáleným přístupem bez nutnosti využívání cloudových služeb. Řešení pak může být umístěno jak v rámci společnosti, tak i v rámci datového centra třetí strany poskytující služby server housingu. Řešení je určeno pro malé a střední podniky, které preferují vlastní nízkoenergetickou a zároveň udržitelnou infrastrukturu s pevně definovanými náklady, před využívání cloudových služeb velkých korporací. Součástí řešení je i konfigurace prostředí pro běh firemní prezentace a definice rozhraní na bázi REST API pro výměnu dat se třetí stranou např. obchodními partnery.

Metodika

Z enterprise distribuce Linuxu se připraví univerzální instalace, která bude doplněna o vlastní balíčky, které nejsou součástí enterprise distribuce. Skripty provedou na jednotlivých Raspberry PI nastavení, které bude odpovídat jejich roli v rámci infrastruktury, tedy připraví jak základní síťové služby, služby VPN, emailový server, databázový server s replikací, server pro správu uživatelů, komunikační server pro chat i video konference, dokumentové úložiště s verzováním a servery pro poskytování webové prezentace a REST API do firmy i třetím stranám. V rámci konfigurace REST API bude vytvořena ukázková definice rozhraní na bázi standardu OpenAPI 3. Služby budou provozovány v režimu vysoké dostupnosti a budou zabezpečeny proti neoprávněnému užití a přetížení. Pro uživatelské stanice se připraví konfigurace prostředí pro práci s poskytovanými službami v rámci firemní sítě a to jak v lokální síti tak i prostřednictvím VPN. Výsledné řešení bude možné naklonovat na více lokalit.

Doporučený rozsah práce

50-60 stran

Klíčová slova

Raspberry PI, Linux, SME, enterprise distribuce, intranet, extranet

---

Doporučené zdroje informací

Adam K. Dean; Linux Administration Cookbook; 2018 Packt Publishing; ISBN 1789342529
Donald Tevault; Mastering Linux Security and Hardening, 2nd Edition; 2020 Packt Publishing; ISBN 1838981772
Harihara Subramanian, Pethuru Raj; Hands-On RESTful API Design Patterns and Best Practices; 2019 Packt Publishing; ISBN 1788992664
Richard Petersen; Red Hat Enterprise Linux 8: Desktops and Administration; 2019 Surfing Turtle Press; ISBN 1949857077
Shaun Thomas , PostgreSQL 12 High Availability Cookbook, Third Edition, 2020 Packt Publishing; ISBN 9781838984854

Předběžný termín obhajoby

2020/21 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 23. 2. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 2. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 30. 03. 2021

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Raspberry Pi jako IT infrastruktura pro malé a střední podniky" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30.3.2021 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Markovi Píckovi, Ph.D. za odbornou a pedagogickou pomoc při zpracování této bakalářské práce, rodině a kolegům za maximální podporu během tvorby této práce a společnosti POSDEE ČESKÁ REPUBLIKA s.r.o. za poskytnutý hardware a síťovou infrastrukturu.

Raspberry Pi jako IT infrastruktura pro malé a střední podniky

Abstrakt

Diplomová práce se zaměřuje na problematiku využití více jednodeskových počítačů Raspberry Pi a enterprise distribuce Linuxu jako IT infrastruktury malé firemní počítačové sítě z pohledu použitelnosti a dostupnosti. V teoretické části jsou probrány služby, které jsou typické pro firemní infrastrukturu. Důraz je kladen na bezpečnost, výkon, vysokou dostupnost a provoz on permise. Na tomto základě je zhodnocen rozsah služeb vhodných k provozu na méně výkonném hardware pro užití v rámci intranetu i služeb poskytovaných v rámci extranetu a internetu. Praktická část se zaměřuje na vytvoření prostředí pro otestování firemních služeb z pohledu výkonu. Testovaní je provedeno na webových službách na bázi REST API. Zároveň je ověřena realizace vybraných IT služeb v kvalitě obvyklé pro podniková prostředí a vysoké dostupnosti.

Výstupem je zhodnocení použitelnosti nasazení Raspberry Pi v rámci infrastruktury podnikové sítě, jak z pohledu výkonu, spotřeby energie, kvality poskytované služby, tak cenové kalkulace pro provoz v období jednoho a tří let.

Klíčová slova: Raspberry Pi, Linux, Ubuntu, REST, OpenAPI, IT infrastruktura, vysoká dostupnost, bezpečnost, intranet, extranet

Raspberry Pi as IT infrastructure for small and middle enterprises

Abstract

The diploma thesis focuses on the issue of using multiple single-board computers Raspberry Pi and enterprise distribution of Linux as an IT infrastructure of a small corporate computer network in terms of usability and availability. The theoretical part discusses the services that are typical of corporate infrastructure. The emphasis is on safety, performance, high availability and on permission operation. On this basis, the range of services suitable for operation on less powerful hardware, for use within the intranet and for services provided within the extranet and the Internet is evaluated. The practical part focuses on creating an environment for testing business services in terms of performance. Testing is performed on web services based on the REST API. At the same time, the implementation of selected IT services in the quality usual for business environments and high availability is verified.

The output is an evaluation of the suitability, including recommendations for the deployment of Raspberry PI within the corporate network infrastructure. Both in terms of performance, energy consumption, quality of service provided and pricing for operation for a period of one and three years.

Keywords: Raspberry Pi, Linux, Ubuntu, REST, OpenAPI, IT infrastructure, high availability, security, intranet, extranet

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Za poslední rok, především pak díky globálnímu dopadu onemocnění COVID-19 byla většina drobných podnikatelů a firem z kategorie malých a středních podniků donucena udělat zásadní změny směrem k digitalizaci své komunikace a firemních procesů. V této kategorii jsou drobní podnikatelé vymezeni jako podnikatelé, kteří zaměstnávají méně než 10 osob a jejichž roční obrat nebo bilanční suma roční rozvahy nepřesahuje 2 miliony EUR [1]. Potřebu změn v tomto segmentu vynutila jednak ekonomická situace, ale současně i legislativní rámec. V souvislosti s opatřeními v boji s pandemií je požadováno po firmách umožnit svým zaměstnancům, pokud je to možné, práci na dálku [2]. V této práci je rozvedeno, jak naplnit technické požadavky na infrastrukturu alternativním způsobem k využívání cloudových služeb.

Vzhledem k naléhavosti této změny většina malých podnikatelů řešila tuto potřebu nejčastěji užíváním cloudových služeb, což je v kontextu snadného zavedení nejjednodušší. Rozhodně nelze tyto služby paušalizovat jako nejlepší, nejlevnější a už vůbec ne jako nejbezpečnější řešení. Většina cloudových služeb, které jsou využívány segmentem malých podnikatelů je mířena na retailového klienta, tedy fyzické osoby. Klíčovým parametrem pro tuto cílovou skupinu je cena služby. Zde je třeba rozlišovat mezi cenou služby, kterou uživatel platí formálně a fakticky. Příkladem budiž služba dokumentového úložiště, které je do určitého objemu zdarma v kontrastu s tím, jaké obchodní podmínky musí uživatel přijmout a tedy faktickou cenu zprostředkovaně zaplatit.

Lze konstatovat, že velké procento SME uživatelů těchto služeb neprovedlo analýzu rizik a vystavují se tedy možné ztrátě nebo postihu v souvislosti s instituty, jako je obchodní tajemství, vlastnictví dat, ochrana osobních údajů nebo přerušení poskytování služby. Tuto problematiku lze posuzovat i v kontextu zájmů třetí strany, jelikož většina takových služeb předpokládá, že uživatel souhlasil s užíváním služby s plnou odpovědností za případnou újmu třetí straně, např. zákazníkovi. Opět bez detailní analýzy rizik a smluvních vztahů nelze toto relevantně vyhodnotit. V tomto kontextu lze tedy konstatovat, že IT služby určené pro retailový segment trhu nejsou ve velké většině případů určené pro užití v rámci SME bez rizik plynoucích z jejich užívání.

Cílem této práce je navrhnout a vytvořit konfiguraci řešení na bázi Raspberry Pi a opensource software pro drobné podnikatele ze segmentu SME. Toto řešení bude schopné po stránce infrastruktury poskytnout rozhraní pro běžné IT služby využívané ve

firemním segmentu. Bude poskytnuta alternativa k privátnímu, hybridnímu nebo veřejnému cloudu, ale i k čistě cloudovým službám poskytovaným jako SaaS. Řešení na bázi Rasperry Pi může sloužit jako alternativní prostředek IT infrastruktury nahrazující cloudové služby nebo též jako doplněk pro využívané cloudové služby firemní klientelou. Velkou výhodou je pak nejenom interoperabilita služeb, ale především pak i nezávislost na poskytovateli služby a tím i odbourání možného vendor lock-in. Vlastní část nebo celá infrastruktura na bázi clusteru Raspberry Pi pak eliminuje riziko plynoucí z nepredikovatelných nákladů. Typická IT infrastruktura, pokud je celá, nebo část poskytována jako cloudová služba, je placena na základě přenosů dat nebo paušálně podle času. V případě Rasperry PI clusteru je pořízení těchto počítačů jednorázový velmi nízký pořizovací náklad a dalším fixním nákladem je pouze konektivita do Internetu a náklady na energie. V případě cloudových řešení obsahují ceníky mnoho položek se závislostmi, které jsou následně na uživatele dynamicky aplikovány dle obchodních podmínek, což se uživatel typicky dozví až zpětně z vyúčtování.

V současné době se stále více hovoří o ESG strategii, tedy strategie environmentální a společenské odpovědnosti a udržitelného rozvoje. Tato strategie se již netýká jen vybraných velkých firem, ale v blízké budoucnosti prakticky všech firem tedy i malých podnikatelů ze segmentu SME. Mít IT infrastrukturu postavenou právě na Raspberry Pi může být pro velké množství firem ze segmentu SME konkurenční výhoda pro dosahování cílů jako je snižování spotřeby energie, uhlíková neutralita, apod.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je vytvořit konfiguraci clusteru z několika Raspberry Pi. Díky enterprise distribuci Linuxu poskytne firmě i klientům zabezpečené vybrané síťové IT služby pro intranet, extranet i internet, jak v rámci vnitřní sítě, tak i vzdáleným přístupem bez nutnosti využívání cloudových služeb. Řešení pak může být umístěno ve více fyzických lokacích. Například v prostorech společnosti nebo v rámci datového centra třetí strany poskytující služby server housingu. Řešení je určeno pro malé a střední podniky, které preferují vlastní nízkoenergetickou a zároveň udržitelnou infrastrukturu s pevně definovanými náklady, před využívání cloudových služeb velkých korporací. V rámci řešení je prezentována i konfigurace vybraných síťových IT služeb a představena definice rozhraní na bázi REST API pro výměnu dat se třetí stranou např. obchodními partnery. V rámci práce je provedeno zhodnocení realizace REST API v kontextu propustnosti na Raspberry Pi.

2.2 Metodika

V teoretické části jsou nejprve diskutovány současné možnosti jednodeskového počítače Raspberry Pi v kontextu aktuálních modelových řad. V rámci práce se rozvádí zejména technické parametry a možnosti užití, rámcově jsou uvedeny i ekonomické parametry. Po technické stránce jsou zdůrazněny možnosti rozšíření lokálního úložiště Raspberry Pi na úroveň, která je obvyklá v podnikovém prostředí v kontextu současných enterprise distribucích Linuxu, ale i v kontextu budoucích rozšíření. V oblasti Enterprise distribucí Linuxu jsou probrány dvě nejrozšířenější, které je možné na architektuře Raspberry Pi provozovat a splňují kritéria pro podnikovou distribuci. Rámcově jsou probrány síťové služby, které mohou být provozovány v rámci intranetu i extranetu a které prakticky každá menší firma nebo podnikatel využívá. Síťové IT služby jsou rozděleny do dvou úrovní a to na nízko úrovňové a vysokoúrovňové. U obou úrovní je v teoretické části popsán rámec a v rámci demonstrativního výčtu pak jsou vybrány služby, které budou v praktické části realizovány.

V praktické části je postupováno dle následujícího scénáře. Je zvolena konkrétní enterprise distribuce Linuxu a provedena instalace image na SD kartu. Následně je tato

distribuce spuštěna na Raspberry Pi 4B+ a provedeno přenesení na SSD NVME disk, včetně změny bootování. Tím je připraveno zařízení, na kterém bude provedena dodatečná manuální konfigurace zařízení a služeb. V další části jsou popsány způsoby konfigurace nízko úrovňových a vysokoúrovňových IT služeb, které budou v rámci této instalace realizovány. Výsledná konfigurace je následně přenesena na další SSD NVME disk pro použití s dalším Raspberry Pi 4B+ a to včetně bootování z tohoto média.

Dále se provede testování a zhodnocení reálného výkonu nízko úrovňových a vysokoúrovňových síťových služeb v kontextu malé firmy v rámci několikatýdenního produkčního provozu. Realizace se provede prostřednictvím volání nízko úrovňových služeb na clusteru relační databáze. V případně vysokoúrovňových síťových služeb pak volání proběhne pomocí CRUD operací REST API, které tvoří v současné době nejčastěji používanou formu komunikace mezi serverem a klientskou aplikací.

Vybrané nízko úrovňové a vysokoúrovňové služby jsou implementovány v režimu vysoké dostupnosti, tedy jsou minimálně duplikované. Výpadek jedné celé instance zcela neochromí chod celého řešení a přepnutí na záložní instanci je pak řešením. V rámci zhodnocení budou kvantifikovány limity a omezení, které s sebou implementace takového řešení na jednodeskové počítače přináší a stanoveny limity vhodnosti užití takového zařízení pro plnění úkolů v rámci intranetu a extranetu, případně prostředí pro běh veřejných aplikací a prezentací u malé firmy.

Platforma Raspberry Pi je primárně podporována na komunitních distribucích Linuxu a i referenční implementace OS je založená na komunitní distribuci Debian. Pro užití v rámci firemní infrastruktury je vhodné použít distribuce Linuxu určené pro podniková prostředí. Distribuce operačního systému Linux, použitá v této práci je Ubuntu 20.04 LTS. Volba této distribuce je založena na definovaných klíčových požadavcích a to stabilitě a délce podpory. Tyto kritéria beze zbytku splnil právě Ubuntu 20 LTS, jelikož podpora této distribuce končí 30. června 2025, zatímco Respberry Pi 4B+ plánuje výrobce uvádět na trh do ledna roku 2025. V praktické části jsou i probrány možnosti přechodu na distribuci založené na RedHat Enterprise Linux.

Na závěr praktické části je provedeno zhodnocení výkonu a naměřených hodnot týkajících se zatížení v průběhu produkčního provozu. Na základě dat o zatížení je určena energetická náročnost provozu a provedena kalkulace nákladů na období jednoho roku a tří let. Vybrané údaje jsou následně srovnány se scénářem, kdy by firma místo Raspberry Pi

realizovala řešení pomocí pronajaté výpočetní kapacity a služeb v cloudu a probrána možnost provozu takové infrastruktury jako nízkoenergetického řešení. Závěrem je provedeno vyhodnocení pilotního provozu REST API a zhodnoceny reálné možnosti Raspberry Pi 4 i v kontextu alternativní architektury x86.

Během tvorby diplomové práce byly využity zdroje uvedené v seznamu literatury.

3 Teoretická východiska

3.1 Hardware z rodiny Raspberry Pi

Raspberry je označení pro produkty britské neziskové organizace Raspberry Pi Foundation, které byly a jsou primárně určeny pro výuku výpočetní techniky a digitálních dovedností. Hardware zahrnuje celou škálu jednodeskových počítačů, od typů vhodných pro jednoúčelová zařízení až po výkonné počítače pro univerzální použití. Právě výkonné jednodeskové počítače jsou použity i v této diplomové práci.

3.1.1 Raspberry Pi 4B

Raspberry Pi 4 Model B je jeden z nejnovějších produktů v řadě počítačů Raspberry Pi, který patří mezi nejvíce používané jednodeskové počítače na světě. Nabízí průlomové zvýšení rychlosti procesoru, multimediálního výkonu, paměti a konektivity ve srovnání s předchozí generací Raspberry Pi 3 Model B+ při zachování zpětné kompatibility a podobné spotřeby energie. Raspberry Pi 4B zachovává stejný půdorys jako starší modely (3B+, 3B, 2B, 1B+), ale přináší významné změny v oblasti konektorů. Napájecí konektor byl povýšen na USB-C, které umožňuje vyšší výkon a pro koncového uživatele poskytuje Raspberry Pi 4 Model B výkon stolního počítače srovnatelný s běžnými PC systémy na bázi procesorů x86. Mezi klíčové vlastnosti tohoto produktu patří následující komponenty [3]:

- vysoce výkonný 64bitový čtyř jádrový procesor Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB, 4GB nebo 8GB LPDDR4-3200 SDRAM (dle zvolené konfigurace)
- dvoupásmová bezdrátová síť LAN 2,4 / 5,0 GHz 2.4 GHz dle IEEE 802.11ac, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2x USB 3.0 porty a 2x USB 2.0 porty.
- podpora duálního zobrazení v rozlišení až 4K prostřednictvím dvojice porty micro-HDMI portů, hardwarové dekódování videa až do H.265 (4kp60 dekódování), H264 (1080p60 dekódování, 1080p30 kódování)
- schopnost PoE (prostřednictvím samostatného doplňku PoE HAT)

Pro účely této práce se použijí modely s 4 a 8 GB RAM. Tento modul je běžně dostupný v cenových relacích od 1 050 Kč za model se 2 GB RAM, přes 1 500 Kč za model se 4 GB RAM až po 2 200 Kč za model s 8 GB RAM. Vhodnost konkrétního modelu je dána případy užití. Model s 2 GB RAM je primárně určen na paměťově nenáročné úlohy s větším využitím CPU, což mohou být např. webové aplikace nebo samostatné infrastrukturní servery. Model s 8 GB RAM se naopak využije v aplikacích, které vyžadují více RAM a méně CPU, což jsou typicky relační databáze. Pro ostatní případy užití se jeví nejvýhodnější užití modelu se 4 GB RAM především pro jeho poměr cena vs. výkon. Raspberry Pi 4B zůstane ve výrobě nejméně do ledna 2026.

Obrázek 1: Rapberry Pi 4B+

Zdroj: raspberrypi.org [3]

3.1.2 Raspberry Pi CM4 IO Board a Raspberry Pi Compute Module 4

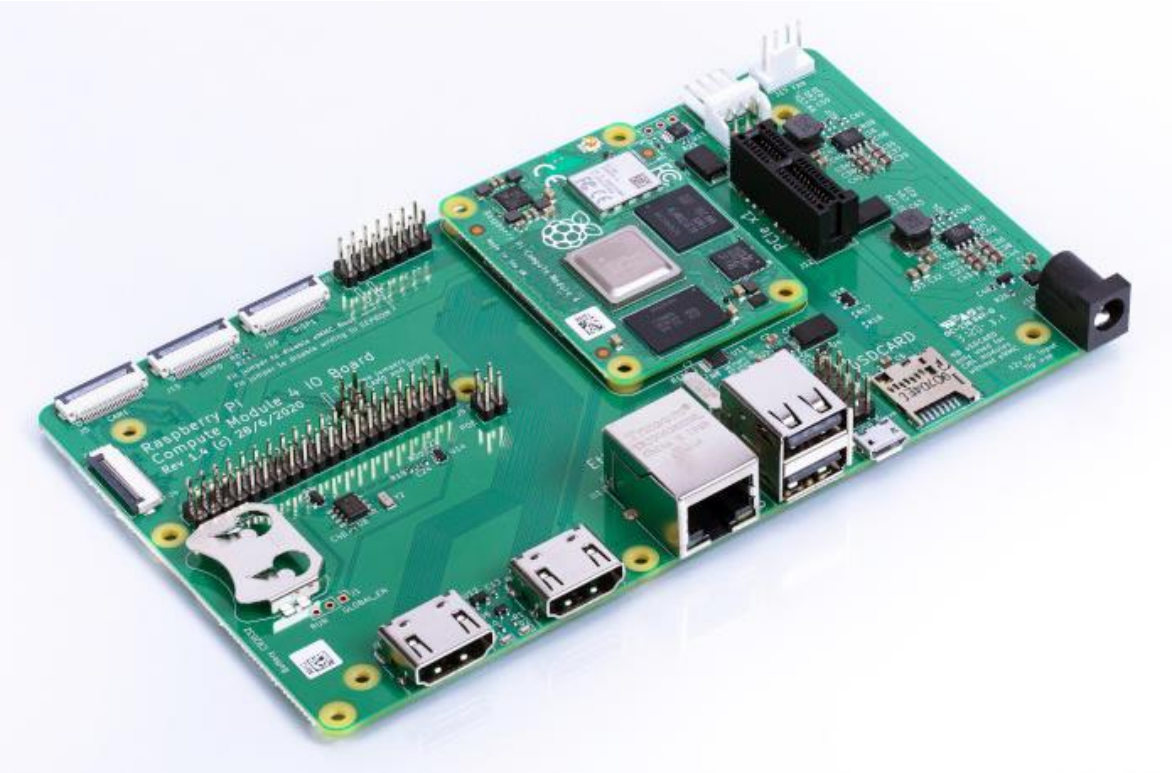
Další z nejnovějších produktů v řadě počítačů Raspberry Pi je Compute Module 4. jedná se o dvojici desek, která dohromady tvoří funkční počítač. K základní desce Raspberry Pi Compute Module 4 IO Board se připojuje Raspberry Pi Compute Module 4. Je navržena jak pro prototypování a vývoj vlastního produktu založeného na CM4, tak jako finální deska integrovaná rovnou do koncových produktů. Je navržena tak, aby vám umožnila rychle vytvářet systémy z běžně dostupných součástek, jako jsou HAT a PCIe

karty, které mohou zahrnovat NVMe, SATA, sítě nebo USB. Hlavní konektory jsou umístěny pro jednoduchost skříně a uživatelský komfort podél jedné strany.

Compute Module 4 (CM4) přináší sílu Raspberry Pi 4 do rodiny výpočetních modulů Raspberry Pi. CM4 je k dispozici v různých variantách podle kapacity eMMC úložiště: 0 GB (Lite); 8GB; 16GB; a 32GB. Nově také nabízí volitelné bezdrátové připojení a výběr kapacity DRAM: 1GB; 2GB; 4GB; a 8GB. Výsledkem je 32 odlišných variant produktu [4]. Raspberry Pi Compute Module 4 IO Board a Raspberry Pi je Compute Module 4 zůstane ve výrobě nejméně do ledna 2028.

Tento produkt v době vzniku této diplomové práce nebyl fyzicky k dispozici, pouze bylo možné jeho předobjednání, nicméně je zde zmiňován v kontextu budoucího možného rozvoje.

Obrázek 2: Rapberry Pi CM4 IO Board a Raspberry Pi Compute Module 4



Zdroj: raspberrypi.org [4]

3.1.3 Rozšiřující hardwarové moduly

Raspebrry Pi 4B disponuje standardně úložištěm v podobě SD karty. Přestože je možné v současnosti pořídit úložiště Micro SDXC o kapacitách 64 resp. 128 GB, kde rychlost čtení dosahuje nejméně 90 MB/s a rychlost zápisu 80 MB/s nelze tyto úložiště doporučit do segmentu SME. Důvodem je jejich omezená spolehlivost, což pro retailové užití není na rozdíl od firemního segmentu klíčový parametr. Pro tyto účely vznikly specializované desky, které je možné kombinovat s Raspberry Pi 4B a připravit tak úložiště vhodná pro firemní segment, tedy i SME.

Jedním z nejlepších na trhu jsou produkty značky Suptronics, které nabízí možnost vytvoření úložiště jak pomocí rotačních disků s rozhrním SATA, tak i pomocí moderních SSD disků na bázi rozhraní mSATA, M.2 NGFF SATA a M.2 NVME. Desky s rozhraním SATA existují ve variantě, kdy je možné jejich stohování, tedy je možné k jednomu Rasperry PI 4 připojit více disků. Naopak pro rozhraní mSATA a M.2 se předpokládá připojení jednoho takového zařízení v rámci rozšiřující desky a vzhledem k počtu USB 3.0 konektorů nelze reálně k zařízení připojit více jak dvě rozšiřující desky bez degradace výkonu.

V současné době rozšiřující desky podporují SSD až do kapacity 2TB a pevné disky do kapacity až 14 GB. V rámci zvolené technologie disků a jejich rozhraní jsou dostupné i různé fyzické velikosti zařízení a to jak SSD, tak i pevných disků.

Obrázek 3: Rapberry Pi 4 s rozšiřující deskou pro SSD NVME Disk

Zdroj: suptronics.com [5]

3.1.4 Raspberry Pi 400

Raspberry Pi 400 je kompletní osobní počítač zabudovaný do kompaktní klávesnice. Raspberry Pi 400 je ideální pro práci s webovými aplikacemi, vytváření a úpravy dokumentů. Raspberry Pi 400 je k dispozici v řadě různých regionálních variant tedy i verze s českou klávesnicí a vše, co je potřeba navíc je jen monitor. Parametry tohoto počítače jsou následující [6]:

- vysoce výkonný 64bitový čtyřjádrový procesor Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
- 4GB LPDDR4-3200 SDRAM
- dvoupásmová bezdrátová síť LAN 2,4 / 5,0 GHz 2.4 GHz dle IEEE 802.11ac, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2x USB 3.0 porty a 1x USB 2.0 port
- podpora duálního zobrazení v rozlišení až 4K prostřednictvím dvojice porty micro-HDMI portů, hardwarové dekódování videa až do H.265 (4kp60 dekódování), H264 (1080p60 dekódování, 1080p30 kódování)

Obrázek 4: Rapberry Pi 400

Zdroj: raspberrypi.org [6]

V případě, že v rámci podnikové IT infrastruktury běží primárně webové aplikace, případně aplikace, které lze provozovat na operačním systému Linux a architektuře ARM jako je např. LibreOffice a není zde závislost na operačním systému Microsoft Windows, lze tento počítač výhodně využít jako klientskou stanici. Z pohledu IT infrastruktury tím dochází nejenom ke zjednodušení správy, ale především i k významnému zvýšení bezpečnosti a to z následujících důvodů:

- Počítač lze uživateli předat již předinstalovaný a to včetně účtů, zabezpečení, firewallu a automatické VPN. Zařízení lze jednoduše vzdáleně spravovat kdekoliv.
- Lze zcela eliminovat možnost připojování vlastních zařízení k počítači např. prostřednictvím USB a v případě nutnosti pak povolit pouze konkrétní zařízení.
- Lze realizovat kompletní auditní stopu v rámci zařízení, tedy je možné na zařízení pracovat s dokumenty, které obsahují např. citlivé informace. Uživatel má v rámci zařízení pouze omezení práva. Vybrané úlohy vyžadující specifická oprávnění lze realizovat prostřednictvím nastavení operačního systému.
- Zařízení je k dispozici za cenu nižší než 2 000,- Kč, tedy významným způsobem nezatíží IT rozpočet žádné malé společnosti nebo podnikatele.
- Webové aplikace musí být kompatibilní s webovými prohlížeči založenými na Chrome (Chrome, Edge, Opera) nebo Firefox

3.2 Podnikové linuxové distribuce

Enterprise distribuce Linuxu je označení pro distribuci Linuxu, která je určena pro podnikové užití. Specifikum podnikového užití spočívá zejména ve stabilitě softwarových produktů v rámci distribuce a délky podpory. Typická délka podpory jedné verze takové distribuce je pak nejméně 5 let ve standardním režimu. Tato doba odpovídá maximálnímu bezpečnému intervalu provozování běžné podnikové aplikace bez zásadních změn, tedy v podmínkách běžné údržby aplikace bez zásadních architektonických změn.

Delší doba provozu přináší vyšší náklady na případný upgrade, pokud je vůbec po delším čase jednoduše proveditelný. Vzhledem k tomu, že Raspberry Pi 4B je produktově garantováno do ledna roku 2026 [3] je taková doba podpory dostačující.

3.2.1 Ubuntu LTS

Ubuntu LTS [7] je rozšířená distribuce OS Linux v podnikovém prostředí. Na rozdíl od CentOS, který vychází z RedHat Enterprise Linuxu a používá tedy balíčkovací systém RPM, je Ubuntu LTS založena na distribuci Debian a její balíčkovací systém je tedy odlišný. Vztah k Raspberry Pi je zde výrazně bližší, jelikož referenční implementace OS pro Raspberry Pi nazvaná Raspberry Pi OS [8] vychází z distribuce Debian. Samotná distribuce Ubuntu je ve verzi 20.04 certifikována [9]. Výhodou Ubuntu LTS provozovaného na Raspberry Pi je velká podobnost s referenční implementací, což může být podstatné především pro začínající uživatele. V neposlední řadě je benefitem i rozsáhlá komunita uživatelů i vývojářů.

Ubuntu LTS má také garantovaný cyklus aktualizací. Pro verzi 20.04 tyto aktualizace skončí 30. dubna 2025 [10]. Oproti CentOS resp. RedHat Enterprise Linux je tedy poloviční [11].

3.2.2 RedHat Enterprise Linux a jeho klony

Binárně kompatibilních distribucí s RedHat Enterprise Linux existuje větší množství, přičemž dominantní na tomto poli je CentOS. Přestože se jedná o komunitní distribuci, právě RedHat měl v jejím aktuálním vývoji zásadní slovo. Společnost IBM po převzetí společnosti RedHat se rozhodla udělat poměrně zásadní změny s velkým dopadem na tuto distribuci. Změny se týkají především procesu uvolňování oprav, který se radikálně změnil [12]. RedHat Enterprise Linux je založený na svobodném a otevřeném software, v souladu s licencí jsou publikovány zdrojové kódy této distribuce. Ty může následně kdokoliv další převzít a publikovat vlastní distribuci. To po oznámení společnosti IBM udělaly např. klony Rocky Linux [13] nebo Alma Linux [14].

RedHat Enterprise Linux podporuje i instalace založené na architektuře ARM, tedy i instalaci na Raspberry Pi. Z distribuce je patrné, že tato architektura je v zájmu této společnosti a to nejen v oblasti jednodeskových počítačů, ale především serverů s procesory ARM. Ty umožňují vytvářet levnější serverovou alternativu oproti technologii

Intel x86 a pro vybrané případy užití jsou tedy způsobem, jak zajistit snižování ceny služeb při zachování kvality.

CentOS, díky tomu, že vychází z distribuce RedHat Enterprise Linux využívá aktualizace programových balíčků. Pro verzi 7 tyto aktualizace skončí 30. června 2024 a nedotýkají se jej změny v modelu distribuce oprav [15]. CentOS verze 8 má stávající podporu pouze do konce rok 2021 a pak je nutné zvolit model dalšího pokračování. Tedy akceptovat podmínky změny distribuce oprav v CentOS nebo přejít na alternativní binárně kompatibilní distribuci.

3.3 Základní síťové služby

Pro prostředí malé firmy je nutné provozovat několik základních síťových služeb. Klíčové hledisko pro členění těchto služeb je jejich dostupnost v rámci vnitřní sítě ať už lokální nebo v rámci VPN nebo z internetu. Služby dostupné pouze z vnitřní sítě, pak naslouchají pouze na vnitřním síťovém rozhraní a přístup odjinud blokuje firewall. Služby dostupné z Internetu, pak typicky běží v tzv. demilitarizované zóně, což je logická podsíť, oddělená od vnitřní sítě a přístup je opět řízen firewallem.

3.3.1 NTP

Podmínkou nutnou v každé počítačové síti je synchronizace času, jelikož úspěšnost mnoha operací je na aktuálním čase přímo závislá. Autorita určující čas je v počítačové síti NTP server [16], který čas pravidelně synchronizuje s hlavními časovými servery. NTP využívá hierarchický systém, kdy nejvýše jsou hodiny synchronizované pomocí hardware např. GPS [48].

3.3.2 DNS

DNS server je aplikace, která provádí předklad doménový jmen na IP adresy a obráceně [16]. Jednou ze zajímavých vlastností DNS serveru je rozkládání zátěže. To je realizováno způsobem, kdy se více IP adresám přidělí stejné jméno. Pokud se takto využívají servery v Internetu, je rozkládání zátěže stochastický proces, protože odpovídají různé servery v různé vzdálenosti. Není tedy dopředu jasné, který server odpoví. V případě lokálních DNS je situace jasně definovaná, jelikož odpovídají stále stejné servery, které postupně záznamy rotují [17].

3.4 Databázový server

Každá softwarová infrastruktura v rámci podnikového prostředí musí obsahovat databázový server. Vzhledem ke zvolené hardwarové platformě a operačnímu systému jsou možnosti výběru redukované. Nicméně nejméně dvě databáze je možné provozovat v režimu vysoké dostupnosti a to MariaDB a PostgreSQL. MariaDB je hojně zastoupena v rámci webových projektů využívajících CMS (WordPress, Joomla nebo Drupal) a díky tzv. LAMP (Linux, Apache, MySQL a PHP). PostgreSQL je pak používána primárně jako univerzální objektově relační databáze pro univerzální podnikové užití, a proto se využívá i v této diplomové práci.

Každá databáze však má jedinečnou sadu funkcionalit a funkcí, které je potřeba zohlednit při konkrétním případu užití. Až donedávna se doporučovalo pro co nejvyšší rychlost provádění dotazů použít MariaDB. Od roku 2019 již toto doporučení neplatí, jelikož rychlost PostgreSQL je srovnatelná a v některých případech i vyšší. MariaDB typicky uzamyká data na úrovni tabulky. PostgreSQL naopak uzamkne data na úrovni řádků. V PostgreSQL je uzamykán pouze aktuální zpracovávaný řádek. Uzamčení na úrovni řádků výrazně snižuje výkonový efekt zamykání v prostředích, kde jsou velké počty současných uživatelů. Obecně je tedy pro PostgreSQL vhodnější prostředí s vysokým zatížením než MariDB. PostgreSQL má procedurální jazyk s názvem PL/pgSQL, zatímco MariaDB ma jiný koncept tzv. „stored procedures“. Z pohledu rozšíření od další programovací jazyky je PostgreSQL lepší volbou [18].

3.4.1 PostgreSQL

PostgreSQL je objektově relačním databázovým systémem s otevřeným zdrojovým kódem pod svobodnou licencí. Běží na všech rozšířených operačních systémech včetně Linuxu, UNIXů a Windows. Plně splňuje podmínky ACID, podporuje cizí klíče, operace JOIN, pohledy, triggery, funkce a uložené procedury. Implementuje většinu standardu SQL92 a SQL99. Je implementována i podpora moderních datových typů jako je JSON nebo XML [19].

3.4.2 Replikace a vysoká dostupnost

PostgresSQL implementuje funkcionality fyzické a logické replikace. Metody fyzické replikace se používají k udržování úplné kopie celých dat jednoho clusteru (v PostgreSQL je cluster sada databází spravovaných jedním hlavním procesem nazvaným postmaster). Zdrojový počítač se označuje jako primary a cíl pak replika. Replika může být provozována ve třech režimech [20]:

- Režim Hot – replika je udržována v reálném čase v co nejaktuálnějším stavu. Z repliky je možné provádět transakce pouze pro čtení.
- Režim Warm – replika je udržována v reálném čase v co nejaktuálnějším stavu. Replika neumožňuje připojení klientů.
- Režim Cold – je obvykle záložní server, který není spuštěný a kde nedochází v reálném čase k udržování dat. Při spuštění jsou nejprve dohrány změny a teprve poté je možné začít obsluhovat požadavky klientů.

Při běžném provozu server PostgreSQL generuje uspořádanou sérii záznamů WAL (write ahead log). Jedná se v o protokol změn, podobný redolog z Oracle nebo binlog z MariaDB. Fyzickou replikací se rozumí transport těchto záznamů na jiný počítač a propagace do běžící databáze repliky. Záznamy WAL se ukládají do stejně velkých souborů, které se nazývají segmenty WAL nebo jen soubory WAL. Staré soubory WAL jsou mazány na základě několika konfiguračních parametrů [21].

PostgreSQL implementuje trigger, který bude vyvolán na primárním serveru při každém vytvoření nového souboru WAL. Tento trigger pak může zkopírovat nový soubor WAL do jiného počítače. Tento postup se nazývá log shipping. Další možnosti je nakopírovat nový WAL soubor např. na NFS nebo do externího úložiště, které se tímto stává archivem. Proces ukládání souborů WAL do archivu se nazývá kontinuální archivace nebo jednoduše archivace WAL. PostgreSQL lze spustit v režimu repliky umístěním platného konfiguračního souboru [22]. V režimu obnovy PostgresSQL importuje a použije pouze soubory WAL generované primárním serverem. Hot a Warm repliky běží v režimu obnovení. Je možné nakonfigurovat načítání souborů WAL z archivu do Postgresu v režimu obnovení.

U replikace v PostgreSQL platí, že je možné ji nakonfigurovat jak v režimu asynchronním tak i synchronním [22]. V případě synchronní replikace dochází

ke dvoufázovému potvrzování na primární databázi, i na replice v rámci jedné transakce. V případě asynchronní replikace dochází k dvoufázovému potvrzování také, ale v rámci transakce se to týká pouze primární databáze. Na replice dochází k dvoufázovému potvrzování mimo transakci. Proto je asynchronní replikace rychlejší a není citlivá na výpadek repliky. Oproti synchronní replikaci ale nelze přepnout primární instanci a repliku okamžitě, ale musí doběhnout proces asynchronní replikace. Existuje zde i velmi malá pravděpodobnost, že by mohlo dojít ke ztrátě transakce při nenadálém výpadku primárního serveru [22].

Logická replikace umožňuje na rozdíl od fyzické, replikovat na úrovni jednotlivých tabulek. Primárně se nepoužívá pro řešení režimu vysoké dostupnosti, ale slouží např. při bez výpadkovém upgrade databáze na vyšší verzi, distribuci zátěže na více serverů, synchronizaci dat mezi databázemi apod [20].

3.5 Správa identit

Identitou v IT se rozumí digitální interpretace skutečné identity z reálného světa v prostředí informačních systémů a počítačových sítí. Skutečnou identitou v tomto kontextu se pak rozumí fyzická osoba, počítač nebo aplikace. V rámci povolení přístupu k vybraným zdrojům dochází ke dvěma krokům a to ověření totožnosti uživatele (označované jako autentizace) a ověření přístupových práv (označované jako autorizace). Z pohledu správy je výhodné mít identity spravované centrálně a současně podporovat i možnost tzv. federované identity. Tím se rozumí možnost výměny informací, které jsou ve správě společnosti i se systémy, které nejsou se společností přímo propojeny. Nejběžnějším způsobem centrální správy účtů jak pak použití adresářové služby na bázi protokolu LDAP.

3.5.1 Adresářové služby a LDAP

Adresářová služba (dále jen adresář) je navržena jako centrální úložiště pro sdílená data a je optimalizována na rychlost vyhledávání a čtení. Taková služba předpokládá data, která se v čase mění velmi málo a současně nevyžadují transakční zpracování. Od adresářové služby se vyžaduje velmi rychlá odpověď, stabilita a dostupnost (jedná se o centrální systém). Data uložená v adresářové službě mají typicky popisný charakter a používají se k definování entity. Typickým příkladem dat, která se uchovávají v adresáři,

jsou identifikační údaje v rámci organizace (např. login, jméno a příjmení uživatele, email, telefon), data aplikací (např. konfigurace jednotlivých komponent nebo služeb), případně informace jmenné služby (např. názvy počítačů, služby, skupiny, uživatelé). Adresářové služby se využívají v mnoha scénářích, pro centrální zpřístupnění informací. Nejčastější klienti adresářové služby jsou operační systémy, systémy pro správu přístupu k aplikacím, webové a mailové servery, relační databáze a další systémy využívané v podnikové infrastruktuře. Nejčastěji používaným protokolem pro komunikaci s adresářovým serverem je LDAP.

LDAP neboli lehký adresářový přístupový protokol je komunikační protokol, který definuje metody pro přístup k adresářové službě. Obecně, LDAP specifikuje způsob, jakým jsou data v adresářové službě reprezentována uživatelům, definuje požadavky na komponenty používané k vytváření datových záznamů v adresářové službě a specifikuje způsob, jakým se k vytváření položek používají základní stavební prvky. LDAP je standardizovaný a otevřený komunikační protokol pro adresářové služby přes TCP/IP. Díky otevřené specifikaci existuje i velké množství jak svobodného tak i proprietárního software, který protokol LDAP implementuje.

Samotná data v systému LDAP se ukládají hlavně v prvcích nazývaných atributy (tzv. Attributes), co jsou typicky páry klíč a hodnota. Atributy se spojují do záznamů (tzv. Entries), která pak tvoří popis konkrétní entity. Záznam je možné považovat za analogii jednoho řádku v relační databázi, Záznamy v LDAP jsou organizovány ve stromové struktuře (Directory Information Tree - DIT). DIT představuje organizační strukturu podobnou systému souborů, kde každý záznam (jiný než záznam nejvyšší úrovně) má přesně jeden nadřazený záznam a může mít pod sebou libovolný počet podřízených záznamů. Nejvyšší záznam v DIT se nazývá root suffix. Záznam v LDAPu se tvoří tři základní prvky [23]:

- Distinguished name (DN) – představuje unikátní název záznamu.
- Atributes - popisují záznam
- Object classes - určují typ záznamu.

Schéma definuje třídy objektu. Každý záznam musí mít atribut objectClass, obsahující pojmenovanou třídu definovanou v adresářovém schématu. Schema lze upravit na míru rozšířením o další atributy. Každý záznam v LDAPu v rámci stromové struktury (DIT) má přiřazené unikátní Distinguished name (DN) aje složeno z atributů a jejich

hodnot. Běžně se využívají standardizované atributy např.. dc (domainComponent), o (organization), ou (organizationalUnit), uid (userid) a cn (common name). První atribut s hodnotou v DN se nazývá relative distinguished name (RDN) ve výše uvedeném příkladu pak RDN obsahuje uid=xkosv302. Řetězec atributů a hodnot popisuje, kde se záznam ve stromové struktuře (DIT) nachází.

V rámci LDAP jsou vztahy mezi daty řešeny pomocí hierarchie, dědičnosti a vnoření. LDAP podporuje jak jednoduchou tak i vícenásobnou dědičnost na úrovni tříd souborem atributů, který je typu MUST (právě jeden) nebo MAY (libovolný počet včetně 0). Ve své definici může objectClass identifikovat nadřazenou objectClass, ze které má zdědit své atributy. Téměř všechny dědičné stromy ObjectClass končí abstraktní třídou ObjectClass zvanou „top“, která je používána k označení vrcholu řetězce dědičnosti.

Typicky je atribut objectClass vícehodnotový a obsahuje třídu top stejně jako čísla dalších tříd. Definice schématu pro každou třídu záznamu vymezuje, jaký druh objektu může záznam reprezentovat. Členství v konkrétní třídě dává záznamu možnost obsahovat alternativní množinu atributů a povinnost obsahovat další povinný soubor atributů.

```
Ukázka DN: uid=xkosv302,ou=people,o=intranet,dc=infok,dc=cz
```

LDAP je protokol, který definuje komunikační rozhraní pro práci s adresářovými službami. V běžné praxi je možné najít následující varianty [23]:

- ldap: // : je základní protokol LDAP, který umožňuje strukturovaný přístup k adresářové službě v nešifrované podobě.
- ldaps: // : je varianta protokolu LDAP, který umožňuje strukturovaný přístup k adresářové službě přes TLS. Je důrazně doporučeno použití tohoto způsobu komunikace, pokud se provozuje protokol LDAP v nezabezpečené i lokální síti síti
- ldapi: // : je varianta, která umožňuje strukturovaný přístup k adresářové službě přes IPC. To se často používá pro bezpečné připojení k lokálnímu systému LDAP pro administrativní účely. Komunikace probíhá přes Unix socket zcela mimo síťovou vrstvu, je tedy možná pouze v rámci jedno počítače.

LDAP uplatňuje čtyři typy ověřování [23]:

- Bez ověření – anonymní přístup, slouží pro přístup do adresáře bez poskytnutí ověřovacích informací. Anonymním uživatelům jsou zpřístupněna pouze data, která nejsou ze své povahy citlivá. Například atributy, obsahující pouze informace o kontaktech.
- Základní ověření - se uskutečňuje na základě znalosti DN (Distinguished Name) a hesla. Data jsou přenášena v textovém formátu, případně zakódována pomocí Base64.
- Ověřování pomocí public key infrastructure (PKI) – se uskutečňuje na základě digitálně podepsaného náhodně vygenerovaného řetězce dat odeslaného spolu s certifikátem na server. Na serveru se provede ověření zaslaného certifikátu.
- Jednoduché ověření s použitím Simple Authentication and Security Layer (SASL) – je ověření některou z alternativních metod poskytovaných tímto frameworkem formou zásuvných modulů např. Kerberos, S/Key, GSSAPI a další

Pro vysokou dostupnost adresářových služeb je nutné využití mechanismu replikace, kdy jsou data z jednoho adresářového serveru automaticky kopírována do jiného adresářového serveru – repliky [24]. V rámci replikace může být kopírována jak stromová struktura jakožto celek, tak i její část. Adresářový server ve funkci master, automaticky kopíruje informace o změnách do ostatních replik. Replikací umožňuje Load balancing tedy rozložení zatížení, zvyšování Fault tolerance a failover, tedy stabilitu a odolnosti proti výpadku, dále pak snižovat dobu odpovědi. Replikace není primárně určena pro rozšiřování možností zápisu do adresáře. Replikace zvyšuje tedy odolnost proti výpadku některého ze serverů. Rozeznáváme tři typy replik [23]:

- Master replika je replika sloužící pro čtení i pro zápis. Obsahuje hlavní kopii adresářových dat.
- Slave replika je podřízenou replikou, která slouží pouze pro čtení záznamů z adresáře.
- Centrální replika je speciální případ slave repliky pro případ více replik v rámci jednoho adresářového serveru, která slouží pouze pro čtení záznamů.

Dle topologie a konfigurace replik existující následující replikace [23]:

- Single master replikace – je základní replikační konfigurace, kde master replika kopíruje data přímo do jednoho nebo více podřízených serverů, které slouží pouze pro čtení. Veškeré změny se provádějí výlučně v master replice.
- Multi-master replikace – je konfigurace, která neobsahuje žádné slave repliky. Master repliky obsahují stejná data, která mohou být upravována na více místech. Následně jsou propagována do ostatních replik.
- Kaskádní replikace – je konfigurace, kdy centrální slave repliky přebírá změny z master repliky a poskytuje je dále slave replice. Centrální slave replika je pouze pro čtení.

Všechny výše uvedené scénáře je možné v rámci topologie dále kombinovat.

3.5.2 Kerberos

Kerberos je síťový autentizační protokol, založený na symetrické kryptografii, který umožňuje bezpečnou oboustrannou autentizaci přes nezabezpečenou síť a je odolný vůči odposlechu a přehrání zachycených zpráv [26]. Standardně se používá k autentizaci v sítích s Windows a UNIX, či přístupu k dalším službám v rámci těchto sítí. Při autentizaci prostřednictvím protokolu Kerberos se po síti nepřenáší heslo a to ani v podobě otisku. Vzhledem ke skutečnosti, že otisk hesla znají obě strany, používá se jako symetrický klíč k šifrování i dešifrování přenášené zprávy mezi klientem a serverem. Celá komunikace probíhá v následujících krocích [27]:

- Uživatel zadá své jméno a heslo. Z hesla se přímo na klientovi vypočítá otisk a na Authentication Server (AS) se odešle požadavek na autentizaci se jménem uživatele.
- AS si ověří existenci uživatele ve své databázi např. LDAP nebo Active Directory. V případě kladného výsledku se použije otisk hesla uživatele jako šifrovací klíč (K1) a zašifruje jím tajný session klíč (K2), kterým bude později šifrována další komunikace s Key Distribution Center (KDC). Kromě tajného session klíče (K2) posílá Authentication Server (AS) na klienta ještě

unikátní klíč Ticket Granting Ticket (TGT), který slouží k identifikaci uživatele a obsahuje jméno uživatele, IP adresu klienta, dobu platnosti ticketu a tajný session klíč (K2). Tento unikátní klíč může dešifrovat jen Ticket Granting Server (TGS). Tím je proces autentizace uživatele dokončen a klient může přistoupit k nějaké serverové službě s podporou Kerberos. Platí, že AS a TGS jsou nedílnou součástí KDC.

V tuto chvíli je už klient autentizován a může se proto pokusit přistoupit k nějaké službě na serveru (S), který podporuje Kerberos dle následujícího schématu [27]:

- Klient posílá na TGS dvě zprávy. První zprávou je TGT s ID služby, ke které chce přistoupit. Druhou zprávou pak tzv. Autenticator (A) obsahující ID klienta a časovou značku, který je šifrován tajným session klíčem (K2).
- TGS dešifruje TGT a tím získá tajný session klíč (K2), kterým může následně dešifrovat zprávu A.

Protokol je odolný proti odposlechu, jelikož komunikace je šifrovaná. Jak plyne z výše uvedeného popisu, nejslabším článkem protokolu Kerberos jsou slabá hesla. Síla hesla je přímo úměrná entropii. Proto je protokol Kerberos určen primárně pro vnitřní sítě. Pro účely autentizace k webovým serverům se používá mechanismus SPNEGO definovaný v RFC-4559. Pro využití na Internetu směrem k federaci identit se využívá v současnosti protokolů SAML2, Oauth2 a OpenID Connect [28].

3.6 Webový server a REST API

V současnosti existují dva nejrozšířenější webové servery a to Apache HTTPd server a NginX. V případě NginX je možné k němu přiřadit i OpenResty, což je pouze rozšířená verze NginX o LuaJIT a další moduly. Zatímco Apache HTTPd [29] je zástupcem klasického webového serveru s rozsáhlou možností konfigurace je NginX stavěn primárně na obsluhu velkého množství požadavků současně (v řádu deseti tisíců a výše) [.

Oba servery zásadním způsobem odlišuje jejich architektura. Apache HTTPd je zástupcem serveru, který využívá mechanismu procesů a threadů pro zpracování požadavků. Každý požadavek zpracuje samostatný thread nebo proces. NginX je vystavěn na událostmi řízené architektuře. Místo zahájení nového procesu nebo nového vlákna pro každý požadavek v události řízené architektuře je tok programu řízen událostmi tedy formou zaslané zprávy z jiného procesu nebo vlákna. Jeden proces běží jako „posluchač“

nebo „detektor událostí“, který čeká na požadavky. Při příchodu požadavku místo zahájení nového procesu „posluchač“ odešle zprávu do jiné části serveru. Ta se nazývá obslužná rutina události a provede daný úkol. Ve výsledku je zapotřebí méně času procesoru a méně paměti. [31]

3.6.1 Poskytovatel statického obsahu

Velká část webového obsahu je statická, tedy jedná se soubory, které se nemění v čase. Typicky se jedná o obrázky, fonty, styly ale i webové stránky nebo webové aplikace psané v jazyce Javascript. Tento obsah je pak výhodné na webovém serveru nabízet tak, že je na straně prohlížeče ukládán v mezipaměti a tak se zvyšuje odezva jednotlivých stránek, díky nižšímu přenosu dat. Role webového serveru je zde především správně řídit obsah a nastavovat http hlavičky tak, aby nedocházelo ke zbytečnému opakovanému načítání stejných dat. Jak Apache HTTPd [29] tak i NginX [30] tuto funkcionalitu podporují.

3.6.2 Reverzní proxy

Proxy server je zprostředkující server, který předává požadavky na obsah z více klientů na různé servery přes internet. Reverzní proxy je typ proxy serveru, který je obvykle umístěn za firewallem v privátní síti a směruje požadavky klientů na příslušný backend server. Reverzní proxy server poskytuje další úroveň abstrakce a kontroly, která zajišťuje plynulý tok síťového provozu mezi klienty a servery. Mezi běžné způsoby použití reverzního proxy serveru patří [32]:

- Rozložení zátěže - reverzní proxy distribuuje požadavky klientů skupině serverů způsobem, který maximalizuje rychlost a využití kapacity a zároveň zajišťuje, že žádný server není přetížen. Pokud server selže, nástroj pro vyrovnávání zatížení přesměruje provoz na zbývající online servery.
- Webová akcelerace - reverzní proxy komprimuje příchozí a odchozí data a ukládá do mezipaměti často požadovaný obsah. To zrychluje tok provozu mezi klienty a servery.
- Zabezpečení backendu - reverzní proxy server funguje jako další ochrana před bezpečnostními útoky. Zajišťuje také, aby bylo možné přistupovat k více serverům bez ohledu na strukturu vaší místní sítě.

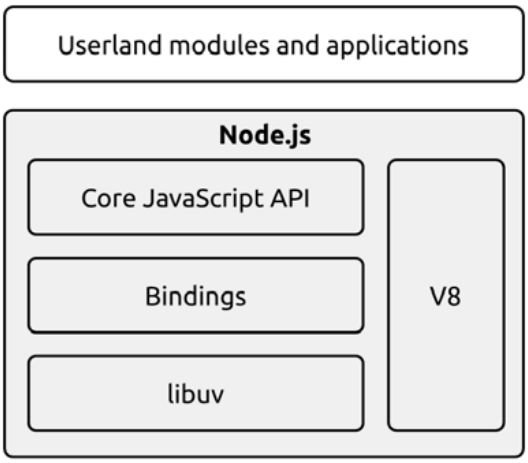
3.6.3 Node.JS

Node.JS je platforma, pro spouštění kódu napsaného v jazyce Javascript. Sdílí technologii pro práci s tímto programovacím jazykem, tedy jádro V8, s prohlížečem Chrome. Jádro Node.JS, tedy běhové prostředí a vestavěné moduly má své základy postavené na několika principech. První princip je malé a stabilní jádro, tedy co nejmenší soubor funkcí. Zbylou funkcionalitu řeší uživatelské moduly. Druhým principem je koncept modulu jako základního stavebního kamene pro vytváření aplikací. V Node.JS je prosazovaný klíčový princip navrhování malých modulů, a to nejen z hlediska velikosti kódu, ale především z hlediska rozsahu. Tento princip má kořeny ve filozofii Unixu zejména ve dvou bodech:

- „Make it easy to write, test, and run programs.“ [36]
- „Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new "features " [37]

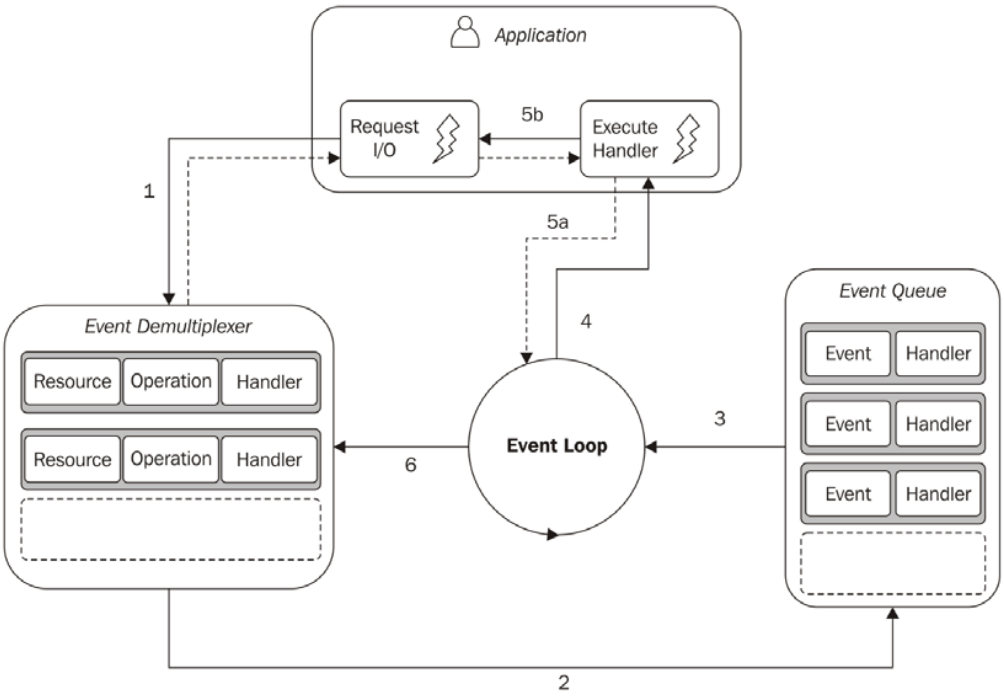
Kromě jasné výhody z hlediska opětovného použití je malý modul také, snadnější pochopení a použití a jednodušší pro testování a údržbu. Princip Don't Repeat Yourself (DRY), tedy neopakování se, je aplikován na zcela nové úrovni. Kromě malé velikosti a rozsahu je žádoucí charakteristikou Node.JS modulu minimalistická sada funkcí vystavená vnějšímu světu na úrovni API, které je jasnější k použití a méně náchylné k chybám používání. Uživatel komponenty zajímá pouze o velmi omezenou sadu funkcí. Charakteristikou modulu je použití, nikoliv rozšíření.

Obrázek 5: Schematické znázornění architektury Node.JS



Zdroj: *Node.js Design Patterns* [34]

Obrázek 6: Princip fungování Node.JS



Zdroj: *Node.js Design Patterns* [34]

1. Aplikace generuje novou I/O operaci odesláním požadavku na Demultiplexer událostí a současně specifikuje obslužnou rutinu, která bude vyvolána po dokončení operace. Odeslání nové žádosti k Demultiplexeru události je neblokující volání a okamžitě se vrátí ovládání do aplikace.
2. Když je sada operací I/O dokončena, Event Demultiplexer vloží sadu odpovídajících událostí do fronty událostí.
3. V tomto okamžiku smyčka událostí iteruje nad položkami fronty událostí.
4. Pro každou událost je vyvolán přidružený obslužný program.
5. Obsluha, která je součástí kódu aplikace, vrací zpět kontrolu po dokončení provádění (5a). Pokud je v rámci obsluhy požadována asynchronní I/O operace (5b), pak je vytvořena další položka, která je přidána do Demultiplexeru událostí (1).
6. Když jsou zpracovány všechny položky ve frontě událostí, smyčka událostí se zablokuje znovu na Demultiplexeru událostí, který poté spustí další cyklus v okamžiku, kdy je k dispozici nová událost.

3.6.4 REST API

REST je architektura distribuovaného integračního rozhraní, které slouží pro jednotný a snadný přístup ke zdrojům prostřednictvím protokolu http [35]. V případě REST se zdrojem rozumí vždy data bez ohledu na to, jakou informaci nesou nebo reprezentují. Na rozdíl od jiných integračních rozhraní jako jsou webové služby na bázi SOAP, REST rozhraní nepodporuje vzdálené volání procedur. Nosná myšlenka této architektury je postavení na tom, že čtyři základní operace s daty tzv. CRUD, tedy vytvoření, načtení, změna a zrušení lze realizovat prostřednictvím čtyř vybraných metod protokolu http tedy POST, GET, PUT a DELETE [38]. Speciálním případem je metoda PATCH, která slouží jako varianta pro partikulární změnu vybraného zdroje.

Přístup REST vede tvůrce rozhraní k tomu, aby uvažoval o datech, tedy provedl jejich modelování, přičemž výčet operací, které může s těmito daty provádět je daný pomocí CRUD operací. [42] Pro usnadnění modelování integračních rozhraní na bázi REST, vzniklo několik specifikací a produktů, které umožňují API rozhraní popsat. V této diplomové práci se využívá specifikace OpenAPI 3.0 [41] (dříve Swagger Specification) [40]. Specifikace API lze psát ve formátu YAML nebo JSON, což jsou snadno čitelné formáty jak pro fyzické osoby, tak i strojové zpracování.

V rámci specifikace rozhraní se definují:

- dostupné koncové body – např. (/ uzivatele) a operace na každém koncovém bodě (GET /uzivatele, POST / uzivatele). Každá jednotlivá operace na koncovém bodě má definovanou unikátní identifikaci v rámci celé specifikace rozhraní. Koncový bod tedy definuje jeden konkrétní datový zdroj.
- datové modely a provozní parametry vstupu a výstupu pro každou operaci definovanou v předchozím bodě
- metody ověřování přístupu k datům
- provozní parametry služby, včetně serverů, licencí apod.

Základní datové typy jsou díky podpoře JSON schématu řetězec, číslo, objekt, pole a prázdná hodnota. [43] V případě řetězce nebo čísla pak existuje ještě upřesnění např. datum pro formát řetězce nebo celé číslo pro typ číslo. Datové typy lze specifikovat blíže pomocí regulárních výrazů, stejně tak je možné detailně definovat kardinalitu v rámci datového modelu. Základní i odvozené datové typy lze využít pro vytvoření popisu

libovolného modelu. V terminologii specifikace se pak hovoří o jednotlivých komponentách, které pak vystupují na vstupu a výstupu u jednotlivých koncových bodů. Po stránce datového modelování specifikace poskytuje pokročilé techniky jako je dědičnost a polymorfismus.

Na základě specifikace API je tedy možné provádět validaci vstupu i výstupu, jelikož komponenty jsou popsány pomocí podmnožiny JSON schématu. Libovolný validátor, který podporuje specifikaci JSON schématu, dokáže ověřit správnost vstupu a výstupu, který je prostřednictvím definovaného API obsluhován. [39] Proto není implementace obsluhy validace vstupu a výstupu nijakým způsobem limitována na konkrétní produkt nebo řešení.

Standardním způsobem následujícího využití specifikace je vygenerování obslužného kódu pro API ve zvoleném programovacím jazyce. Výsledný výstup je dotvořen programátory do podoby finálního produktu, který je vhodný pro nasazení na server. Tento přístup je naprosto dominantní, přestože přináší nižší efektivitu, zejména pokud je potřeba specifikaci měnit v čase. Méně rozšířeným způsobem je běhové zpracování vlastní specifikace. Program pak interpretuje vlastní specifikaci, tedy sestaví pravidla pro jednotlivé obsluhované zdroje a jejich validaci přímo v paměti počítače a jednotlivá volání pak využívají služeb tohoto programu. Výhodou tohoto přístupu je fakt, že při změně specifikace stačí pouze dodat novou specifikaci, nevýhodou pak skutečnost, že takový program je potřeba vytvořit, což je složitější, než jednoduché úpravy do vygenerovaného kódu. V případě, že se použije vybraná existující implementace takového programu, je potřeba mít na paměti případná omezení.

Specifikace OpenAPI [41] umožnila libovolné REST API obohatit o jeden klíčový prvek, který v samotném REST principiálně chybí. Jak bylo popsáno dříve, REST pracuje výlučně se zdroji tedy daty a žádným způsobem nedefinuje, kdo data poskytuje nebo zpracovává. To je ve výlučné kompetenci programátora při implementaci takového rozhraní. OpenAPI navíce definuje u každé operace unikátní položku „operationId“, která je typu text, nicméně její obsah je výlučně na tvůrci konkrétní API specifikace. V případě, že se ze specifikace generuje kód, daná položka má pouze informativní význam. Jiná situace nastává v případě, že je specifikace interpretována za běhu programem. Je možné pomocí této položky řídit i vlastní zpracováním jelikož řetězec je nejenom text, ale i kus programového kódu.

OpenAPI specifikace je publikována pod svobodnou licencí a předpokládá se, že se jedná o rámec, nikoliv konečnou a nerozšířitelnou definici. Pokud nestačí její aktuální forma, pak je dovoleno a doporučeno provést její rozšíření pomocí postupu definovaného ve specifikaci. Výsledná specifikace API, která je kompatibilní s OpenAPI [41], bude stále čitelná pro všechny, zatímco rozšíření budou využitelná pouze pro ty, kdo je implementují nad rámec běžného standardu.

4 Vlastní práce

4.1 Způsob realizace vlastní práce

Realizace vlastní práce vychází z metodiky. Nejprve se provede kompletace hardware, tedy tří kusů Raspberry PI 4B, doplňkových hardwarových modulů a SSD disku. Poté se provede prvotní instalace zvolené distribuce Ubuntu 20.04 LTS na SD kartu včetně následných aktualizací. Následně se systém přenese na disk SSD NVMe a provede se upgrade Raspberry Pi 4B pro bootování ze SSD disku. Tím se připraví zařízení pro další konfiguraci služeb. Tato operace se provede pro celkem tři zařízení Raspberry Pi 4B.

Po instalaci hardware a operačního systému se provede instalace nízko a vysokoúrovňových služeb. Postupuje se podle schématu, kdy nejprve se provede vlastní instalace software na všech třech zařízeních a poté se provede konfigurace a integrace. Konfigurace je započata vždy u prvního zařízení, které je konfigurováno jako master. Ostatní zařízení jsou konfigurovány podle rolí, které v rámci řešení zastupují. Po ukončení konfigurace a integrace je k dispozici celý cluster pro provedení testů.

Celý cluster slouží v rámci pilotního provozu a poskytuje vybrané skupině devíti osob níže popsané služby. Samotná zátěž od fyzických osob je sporadická, faktická zátěž se provede pomocí opakovaných volání CRUD operací REST API z testovacího zařízení v lokální síti, ale mimo tento cluster. Touto realizací se uskuteční jak vysokoúrovňové volání prostřednictvím HTTP protokolu, tak i nízko úrovňových služeb na clusteru relační databáze. Dále se změří spotřeba energie v závislosti na zatížení Raspberry Pi 4B.

Nad clusterem se provede simulace výpadku formou odpojení jednoho Raspberry Pi 4 od počítačové sítě a ověření případných dopadů na úroveň poskytování služeb.

Naměřená data zátěže i doby odezvy se uloží vzhledem k objemu v relační databázi a zde se provede i jejich agregace. Výsledné hodnoty se porovnají s referenční implementací na platformě x86, interpretují a vyhodnotí se vhodnost Raspberry Pi 4B pro využití jako IT infrastruktura pro malé firmy. Dále se vyhodnotí vysoká dostupnost a její praktická použitelnost v rámci clusteru tří Raspberry Pi4B.

4.2 Předpoklady a požadavky

Předpokladem řešení je kompletace serverové jednotky na bázi počítače Raspberry Pi 4B a to nejméně tří kusů pro simulaci serverového clusteru a malá počítačová síť obsahující switch, kde bude tento cluster připojen. Dále pro provedení práce bude do počítačové sítě připojen klient na bázi počítače Raspberry Pi 400, který bude simulována zátěž clusteru pomocí REST API. Pro srovnání bude nainstalován jednodeskový počítač ODROID-H2+ x86 [56] s distribucí Ubuntu 20.04 LTS pro architekturu x86_64. Provozní parametry tohoto jednodeskového počítače jsou nastaveny podobně jako u serverových jednotech Raspberry Pi.

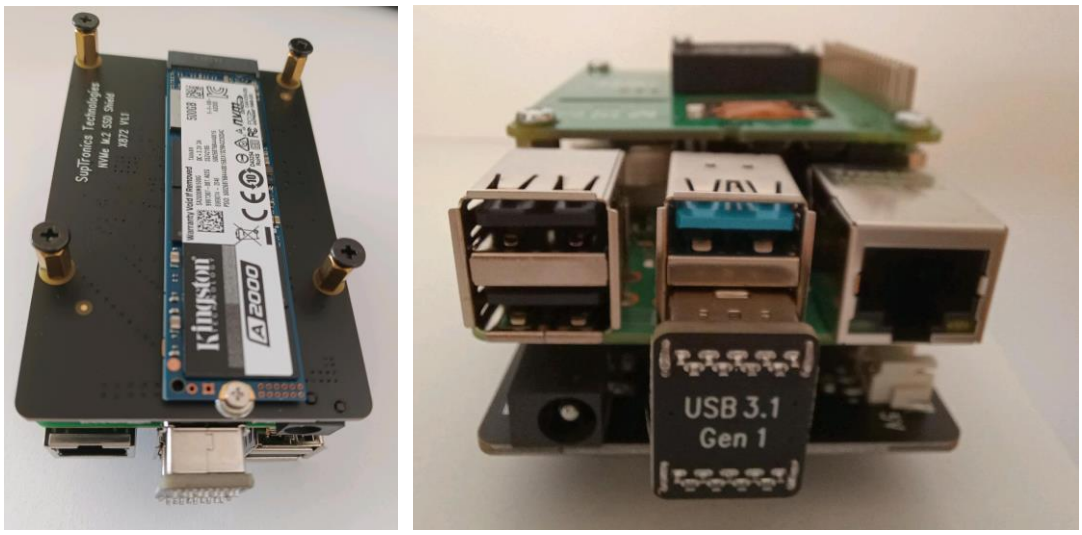
4.2.1 Realizovaná infrastruktura

Praktická realizace Raspberry Pi 4B s dodatečnými moduly je vyobrazena na obrázcích níže. Technické parametry zařízení jsou uvedeny v tabulce. Vlastní testovací infrastruktura se skládá z:

- 3x zkompletovaného Raspberry PI 4
- Raspberry Pi 400, které simuluje klienty
- Switche TP Link TP-SG1008P

Mimo vlastní infrastrukturu, která se neúčastní přímo testů, se zahrnuje počítač pro sběr logů. Po ukončení modulů Raspberry Pi 4B, se infrastruktura využila pro otestování ODROID-H2+.

Obrázek 7: Sestavený modul Raspberry Pi 4 s doplňky



Zdroj: Vlastní zpracování

Tabulka 1: Hardwarové komponenty jednotky Raspberry Pi 4

Č. komponenty	Název	Popis
1	Raspberry Pi 4B	Každé Raspberry obsahuje 8 GB RAM
2	Raspberry Pi PoE HAT	Modul pro napájení prostřednictvím Ethernetu
3	Suptronics X873 V1.2 M.2 SSD Shield	Modul pro instalaci NVMe SSD disku
4	Kingston SSD A2000	SSD disk M.2 (PCIe 3.0 4x NVMe

Zdroj: Vlastní zpracování

4.2.2 Požadované služby

V rámci clusteru je zvolen takový mix služeb, který umožní Raspberry Pi plnit úlohu dvou typů serverů. První typ demonstruje především možné případy užití v malé firmě, druhý typ slouží pro měření propustnosti webových služeb na bázi REST API. Do prvního typu serverů se zahrnují:

- Server pro správu verzí Gitea
- Emailový ekosystém složený z programů Postfix a Dovecot
- Server pro online spolupráci EtherPad Lite

Do druhého typu serverů se zahrnují:

- Adresářový server OpenLDAP s multi master replikací
- Databázový server PostgreSQL s replikací
- Webový server OpenResty a prostředí Node.JS pro běh software Aengine

4.3 Instalace a konfigurace Raspberry Pi

Raspberry Pi 4B je nejprve zkompletováno do finální podoby, tedy včetně SSD disku, který obsahuje obraz serverové distribuce Ubuntu 20.04 LTS a modulu podporujícího Power over Ethernet. Výsledný produktu je představen na obrázku níže. Následně je zařízení otestováno, zda bootuje z SD karty serverovou distribuci Ubuntu 20.04 LTS.

4.3.1 Upgrade bootu SD karty na SSD disk

Vlastní upgrade bootu z SD karty na SSD disk je provedena ve dvou krocích. V prvním kroku se provede aktualizace firmware Raspberry Pi [50]. Neprve je potřeba provést kontrolu souboru sudo /etc/default/rpi-eeprom-update, zda obsahuje:

```
FIRMWARE_RELEASE_STATUS="stable"
```

Poté je možné provést aktualizaci firmware a nezbytné restartovat počítač.

```
sudo apt update && sudo apt full-upgrade -y
sudo rpi-eeprom-update -a
```

Po startu počítače se zkontroluje přítomnost zařízení /dev/mmcblk0 reprezentující SD kartu a /dev/nvme0 reprezentující SSD NVMe disk. Poté je spuštěna sada příkazů. V rámci volání skriptu je provedeno především nastavení správných parametrů v config.txt a přidána automatická dekomprese jádra.

```
#/bin/bash
sudo umount /media/pi/writable
sudo umount /media/pi/system-boot
sudo mkdir /mnt/boot
sudo mkdir /mnt/writable
sudo mount /dev/nvme0p1 /mnt/boot
sudo mount /dev/nvme0p2 /mnt/writable
sudo curl https://raw.githubusercontent.com/TheRemote/Ubuntu-Server-raspi4-unofficial/master/BootFix.sh | sudo bash
chmod +x BootFix.sh
sudo ./BootFix.sh
sudo umount /mnt/boot
sudo umount /mnt/writable
```

Po úspěšném proběhnutí je možné SD Raspberry Pi 4 vypnout a SD kartu vyjmout. Proces boot se provede výhradně pomocí SSD disku.

4.4 Implementované služby

Na začátku implementace je vybráno několik služeb, které slouží jako infrastruktura pro poskytované IT služby. Emailový server, server pro správu verzí Gitea a server pro spolupráci na dokumentech EtherPad slouží pro běžnou firemní agendu a pouze demonstrují škálu případů užití Raspberry Pi. Samy o sobě je jejich přínos k zátěži clusteru Raspberry Pi zanedbatelný. Adresářová služba OpenLDAP a Kerberos slouží pro ověřování uživatelů a spolu s databází PostgreSQL tvoří jádro IT infrastruktury provozované ve vysoké dostupnosti. REST API implementované pomocí webového serveru OpenResty, aplikačního serveru Node.JS a databáze PostgreSQL ja pak reálná aplikace, která může běžet v podnikovém prostředí a generuje skutečnou kontinuální zátěž.

4.4.1 Emailový server

Emailový server je softwarový ekosystém, složený ze tří softwarových produktů. První je produkt Postfix [46], zastupující roli Mail Transfer Agent (MTA). Rolí MTA je přenášet e-maily odeslané klienty jiným MTA a přijímat e-maily od jiných MTA, které se uloží v poštovním adresáři uživatele. Druhým produktem je Dovecot [51], který poskytuje obsah poštovního adresáři uživateli prostřednictvím protokolu IMAP. Dovecot je nasazen také v roli poskytovatele autentizace SASL pro Postfix. Třetím produktem je LDAP server, poskytující informace o uživatelských účtech a e-mailových aliasech. V programu Postfix je využíváno virtuálních poštovních schránek, bez ohledu na skutečnost, zda daný uživatel existuje v kontextu operačního systému.

V rámci adresáře LDAP je vyžadováno uchovávat uživatelské jméno a heslo pro ověření, UID a GID pro správu oprávnění poštovního adresáře uživatele, umístění poštovního adresáře a seznam poštovních aliasů pro daného uživatele.

Instalace Postfix i Dovecot se provede standardním mechanismem instalace balíčků. Před započetím konfigurace Postfix pro příjem a odesílání pošty se vytvoří několik vyhledávacích tabulek LDAP, které postfix použije k dotazování na adresář. Pro zvýšení bezpečnosti je převeden adresář /var/spool/postfix do tzv. chroot prostředí. Poté se provede konfigurace Postfix. V další roku je provedena konfigurace IMAP serveru Dovecot včetně SASL metody autentizace. Následně se provede konfigurace dalších služeb emailového ekosystému zahrnující antispamovou a antivirovou kontrolu a přístup k emailu pomocí webového rozhraní. V následujícím textu je představen rámec konfigurace, detailní popis

nastavení emailového ekosystému je možní získat v rámci dokumentace produktů [46], [51].

Produkt Postfix je instalován standardním způsobem z repozitářů.

```
    $ sudo apt-get install postfix postfix-ldap
    $ sudo ldapadd -ZZ -x -W -D cn=admin,cn=config -H ldapi:// -f
postfix.ldif
```

V rámci nastavení Postfix se provede mapování pro LDAP. Jednotlivé soubory jsou umístěny v /etc/postfix/ldap včetně nastavení zabezpečení přístupu k těmto souborům. V konfiguraci Postfix je pak mapování odkazováno.

```
    # Společné připojení pro všechny LDAP mapování uvedení dále
    server_host = ldapi://
    bind = yes
    bind_dn = cn=mailAccountReader,ou=Manager,dc=dp2021,dc=online
    bind_pw = __HESLO__
    search_base = ou=Mail,dc=dp2021,dc=online
    scope = sub

    # /etc/postfix/ldap/virtual_alias_domains
    query_filter = mailacceptinggeneralid=*@%s
    result_attribute = mailacceptinggeneralid
    result_format = %d

    # /etc/postfix/ldap/virtual_alias_maps
    query_filter = mailacceptinggeneralid=%s
    result_attribute = maildrop

    # /etc/postfix/ldap/virtual_mailbox_maps
    query_filter = maildrop=%s
    result_attribute = homeDirectory
    result_format = %s/mailbox/

    # /etc/postfix/ldap/virtual_uid_maps
```

```
query_filter = maildrop=%s
result_attribute = uidNumber
smtpd_sender_login_maps
query_filter = (|(mailacceptinggeneralid=%s)(maildrop=%s))
result_attribute = uid

$ sudo chown postfix:postfix /etc/postfix/ldap/*
$ sudo chmod 400 /etc/postfix/ldap/*
```

V rámci nastavení Postfix se provede nastavení konfigurace emailového serveru v konfiguračním souboru /etc/postfix/main.cf.

```
myhostname = server1.dp2021.online

virtual_alias_domains = ldap:/etc/postfix/ldap/virtual_alias_domains
virtual_mailbox_domains = $myhostname

virtual_alias_maps = ldap:/etc/postfix/ldap/virtual_alias_maps
virtual_mailbox_base = /
virtual_mailbox_maps = ldap:/etc/postfix/ldap/virtual_mailbox_maps
virtual_uid_maps = ldap:/etc/postfix/ldap/virtual_uid_maps
virtual_gid_maps = ldap:/etc/postfix/ldap/virtual_uid_maps

$ sudo chown mkdir -p /home/mail
$ sudo chown chown :mail /home/mail && chmod g+w /home/mail
```

Také produkt Dovecot je instalován standardním způsobem z repozitářů.

```
$ sudo apt-get install dovecot-core dovecot-imapd dovecot-ldap
```

V rámci konfigurace a integrace je uvedena konfigurace použití LDAP.

```
passdb {
        driver = ldap
        args = /etc/dovecot/dovecot-ldap.conf.ext
```

```
}
userdb {
        driver = ldap
        args = /etc/dovecot/dovecot-ldap.conf.ext
}
uris = ldapi://
dn = cn=mailAccountReader,ou=Manager,dc=dp2021,dc=online
dnpass = <dn bind password>
debug_level = 0
auth_bind = yes
auth_bind_userdn = uid=%u,ou=Mail,dc=dp2021,dc=online
ldap_version = 3
base = ou=Mail,dc=dp2021,dc=online
scope = subtree
user_attrs = homeDirectory=home,uidNumber=uid,gidNumber=gid
user_filter = (&(objectClass=posixAccount)(uid=%u))
```

Dovecot vystupuje v roli SASL providera pro LDAP a tento autentizační mechanismus je používán v Postfix pro autentizace uživatele při odesílání emailu tzv. relay. Autentizace je provedena mechanismem Unix socketu.

```
# Úprava v /etc/dovecot/conf.d/10-master.conf

service auth {
    ...
    unix_listener /var/spool/postfix/private/auth {
        mode = 0660
        user = postfix
        group = postfix
    }
}

# Úprava nastavení Postfix v /etc/postfix/main.cf.

smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
```

```
    smtpd_sasl_auth_enable = yes
    broken_sasl_auth_clients = yes

    smtpd_relay_restrictions             =             permit_mynetworks
permit_sasl_authenticated defer_unauth_destination
    smtpd_recipient_restrictions = reject_sender_login_mismatch
```

4.4.2 Adresářový server OpenLDAP a Kerberos

Adresářový server OpenLDAP je určený pro uchování informací nezbytných pro autentizaci a autorizaci. V této práci se vychází ze standardních balíčků v rámci distribuce Ubuntu. Celý proces probíhá v několika krocích. Nejprve se provede instalace a konfigurace OpenLDAP serveru na všech nodech. Poté se provede instalace a implementace Kerberos a v posledním kroku se změní konfigurace OpenLDAP tak, aby podporoval multi master replikaci mezi jednotlivými počítači.

Je provedena vlastní instalace OpenLDAP serveru, nastaveno heslo administrátora a po instalaci se provede finální kontrola nastavení. Poté je provedeno přidání uživatelů a skupin.

```
    $ sudo apt -y install slapd ldap-utils
    $ sudo slapcat
    $ sudo cat <<EOF > base.ldif
    dn: ou=people,dc=srv,dc=world
    objectClass: organizationalUnit
    ou: people

    dn: ou=groups,dc=srv,dc=world
    objectClass: organizationalUnit
    ou: groups
    EOF
    $ sudo ldapadd -x -D cn=admin,dc=srv,dc=world -W -f base.ldif
    $ sudo rm –rf base.ldif
```

Na serveru nakonfiguruje v rámci zabezpečení i připojení pomocí TLS.

```
    $ sudo cp /etc/ssl/private/server.key \
    /etc/ssl/private/server.crt \
    /etc/ssl/certs/ca-certificates.crt \
```

```
    /etc/ldap/sasl2/

    $ sudo chown openldap. /etc/ldap/sasl2/server.key \
    /etc/ldap/sasl2/server.crt \
    /etc/ldap/sasl2/ca-certificates.crt

    $ sudo cat <<EOF > mod_ssl.ldif
    dn: cn=config
    changetype: modify
    add: olcTLSCACertificateFile
    olcTLSCACertificateFile: /etc/ldap/sasl2/ca-certificates.crt
    olcTLSCertificateFile: /etc/ldap/sasl2/server.crt
    olcTLSCertificateKeyFile: /etc/ldap/sasl2/server.key
    EOF

    $ sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f mod_ssl.ldif
    $ sudo systemctl restart slapd
```

Dále se v OpenLDAP provede příprava na následnou instalaci Kerberosu. Jsou nainstalovány potřebné balíčky, extrahováno Kerberos schéma, převedeno do formátu ldif a importováno do OpenLDAP. Následně jsou indexovány často používané položky, vytvořeny Kerberos záznamy pro administraci a nastaveny přístupové seznamy.

```
    $ sudo apt install krb5-kdc-ldap krb5-admin-server schema2ldif
    $     sudo     cp     /usr/share/doc/krb5-kdc-ldap/kerberos.schema.gz
/etc/ldap/schema/
    $ sudo gzip -d /etc/ldap/schema/kerberos.schema.gz
    $ sudo ldap-schema-manager -i kerberos.schema
    $ sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// <<EOF
    dn: olcDatabase={1}mdb,cn=config
    add: olcDbIndex
    olcDbIndex: krbPrincipalName eq,pres,sub
    EOF

    $ ldapadd -x -D cn=admin,dc=dp2021,dc=online -W <<EOF
    dn: uid=kdc-service,dc=dp2021,dc=online
    uid: kdc-service
    objectClass: account
    objectClass: simpleSecurityObject
```

```
userPassword: {CRYPT}x
description: Account used for the Kerberos KDC
```

```
dn: uid=kadmin-service,dc=dp2021,dc=online
uid: kadmin-service
objectClass: account
objectClass: simpleSecurityObject
userPassword: {CRYPT}x
description: Account used for the Kerberos Admin server
EOF

$ sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// <<EOF
dn: olcDatabase={1}mdb,cn=config
add: olcAccess
olcAccess: {2}to attrs=krbPrincipalKey
  by anonymous auth
  by dn.exact="uid=kdc-service,dc=dp2021,dc=online" read
  by dn.exact="uid=kadmin-service,dc=dp2021,dc=online" write
  by self write
  by * none
-
add: olcAccess
olcAccess: {3}to dn.subtree="cn=krbContainer,dc=dp2021,dc=online"
  by dn.exact="uid=kdc-service,dc=dp2021,dc=online" read
  by dn.exact="uid=kadmin-service,dc=dp2021,dc=online" write
  by * none
EOF
```

Tím se ukončí základní konfigurace OpenLDAP a je možné přikročit k instalaci a konfiguraci Kerberos.

```
$ sudo apt-get install krb5-kdc
$ sudo dpkg-reconfigure krb5-kdc
```

V rámci konfigurace se provedou změny souboru /etc/krb5.conf.

```
[realms]
 DP2021.ONLINE = {
```

```
            kdc = server1.dp2021.online
            kdc = server2.dp2021.online
            admin_server = server1.dp2021.online
             default_domain = dp2021.online
             database_module = openldap_ldapconf
    }

    [dbdefaults]
     ldap_kerberos_container_dn = cn=krbContainer,dc=dp2021,dc=online

    [dbmodules]
     openldap_ldapconf = {
            db_library = kldap
            disable_last_success = true
            disable_lockout  = true
            ldap_kdc_dn = "uid=kdc-service,dc=dp2021,dc=online"
            ldap_kadmind_dn = "uid=kadmin-service,dc=dp2021,dc=online"
            ldap_service_password_file = /etc/krb5kdc/service.keyfile
                ldap_servers = ldapi:///
                ldap_conns_per_server = 5
       }
```

Pouze na prvním LDAP serveru se provedou následující příkazy pro vytvoření tzv. realmu a přístupových práv. Ostatní servery již využijí mechanismus repliky.

```
    $ sudo kdb5_ldap_util -D cn=admin,dc=dp2021,dc=online create -
subtrees dc=dp2021,dc=online -r DP2021.ONLINE -s -H ldapi:///
    $ sudo kdb5_ldap_util -D cn=admin,dc=dp2021,dc=online stashsrvpw -f
/etc/krb5kdc/service.keyfile uid=kdc-service,dc=dp2021,dc=online
    $ sudo kdb5_ldap_util -D cn=admin,dc=dp2021,dc=online stashsrvpw -f
/etc/krb5kdc/service.keyfile uid=kadmin-service,dc=dp2021,dc=online

    $ sudo echo „*/admin@EXAMPLE.COM       *"  >> /etc/krb5kdc/kadm5.acl
    $ sudo systemctl start krb5-kdc.service krb5-admin-server.service
```

Pro nastavení multi master replikace OpenLDAP na všech serverech nastavíme modul syncprov pomocí následujících příkazů.

```
    $ sudo cat <<EOF > mod_syncprov.ldif
    dn: cn=module,cn=config
```

```
objectClass: olcModuleList
cn: module
olcModulePath: /usr/lib/ldap
olcModuleLoad: syncprov.la
EOF

$ sudo ldapadd -Y EXTERNAL -H ldapi:/// -f mod_syncprov.ldif

$ sudo cat <<EOF >  syncprov.ldif
dn: olcOverlay=syncprov,olcDatabase={1}mdb,cn=config
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov
olcSpSessionLog: 100
EOF

$ sudo ldapadd -Y EXTERNAL -H ldapi:/// -f syncprov.ldif
```

Následně je na všech replikách použita následující šablona, přičemž se mění pouze obsah proměnných *olcServerID* and *provider*.

```
$ sudo cat <<EOF > master01.ldif
dn: cn=config
changetype: modify
replace: olcServerID
olcServerID: __UNIKATNI_CISLO_PRO_KAZDY_SERVER__

dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcSyncRepl
olcSyncRepl: rid=001
  provider=ldap://__IP_JINEHO_MASTERU__:389/
  bindmethod=simple
  # own domain name
  binddn="cn=admin, dc=dp2021,dc=online"
  credentials=password
  searchbase=" dc=dp2021,dc=online"
  scope=sub
```

```
      schemachecking=on
      type=refreshAndPersist
      retry="30 5 300 3"
      interval=00:00:05:00
    -
    add: olcMirrorMode
    olcMirrorMode: TRUE

    dn: olcOverlay=syncprov,olcDatabase={1}mdb,cn=config
    changetype: add
    objectClass: olcOverlayConfig
    objectClass: olcSyncProvConfig
    olcOverlay: syncprov
    EOF
    $ sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f master01.ldif
```

Na závěr je možné nastavit na všech replikách v konfiguračním souboru /etc/ldap.conf následující.

```
    uri ldapi:// ldap://__ADRESA_JINE_REPLIKY__
```

4.4.3 Databázový cluster PostgreSQL

Databázový server PostgreSQL se instaluje v posloupnosti několika kroků. Nejprve se nainstaluje primární databázový cluster a je provedena jeho konfigurace. V dalších krocích jsou postupně klonovány repliky, přičemž na druhém serveru je tzv. hot replika, tedy taková forma repliky, která je udržována aktuální a odpovídá na všechny dotazy na čtení.

Pro instalaci PostgreSQL se použijí balíčky přímo od PostgreSQL, nikoliv ty, které jsou součástí distribuce. V rámci distribuce jsou balíčky vždy o jednu až dvě major verze starší. Je pak reálný předpoklad, že po stránce výkonu jsou balíčky od PostgreSQL lepší.

```
    $ echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -
cs`-pgdg main" | sudo tee  /etc/apt/sources.list.d/pgdg.list
    $ sudo apt-get install postgresql-13 postgresql-client-13
    $ systemctl restart postgresql.service && systemctl status
postgresql.service
```

Po instalaci se provede přepnutí na uživatele *postgres* a je provedeno nastavení primární databáze. Tedy založení datbáze, uživatelů a parametrů pro připojení.

```
    $ sudo su - postgres
    $ psql -c "alter system set wal_level = 'logical'"
    $ psql -c "alter system set listen_addresses = 'localhost,
192.168.10.10'"
    $ psql -c "alter system set max_connections = 100"
    $ psql -c "alter system set password_encryption = 'scram-sha-256'"
    $ psql -c "select pg_reload_conf()"
    $ psql -c "create user replicator with replication"
    $ psql -c "create user deploy"
    $ psql -c "create user dp2021"
    $ psql -c "select pg_reload_conf()"
    $ createdb --encoding=UTF8 --locale=cs_CZ.UTF-8 --template=template0
-p 5432 dp2021
    $ psql -c "grant all privileges on database dp2021 to deploy"
```

V dalším kroku se provede nastavení zabezpečení databáze a možnost přístupu.

```
    $ cat <<EOF >> 13/dp2021/pg_hba.conf
    local   dp2021          dp2021                          peer map=dp
    local   all             all                     peer
    hostssl replication     replicator              192.168.10.11/32
cert map=dbmap clientcert=1
    hostssl replication     replicator              192.168.10.12/32
cert map=dbmap clientcert=1
    hostssl all             deploy                  192.168.10.0/24
cert map=dbmap clientcert=1
    EOF
    $ cat <<EOF >> 13/2021/pg_ident.conf
    dp              node                    dp2021
    dp              postgres                postgres
    dbmap            /^(.*)\.dp2021\.online$       \1
    EOF
    $ psql -c "select pg_reload_conf()"
```

Realizace replika PostgreSQL na každé další instanci se provede posloupností následujících příkazů. Pokud se jedná o *hot* repliku, je tato ihned po vytvoření spuštěna. Naopak u *cold* repliky není okamžité spuštění vyžadováno.

```
    $ echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -
cs`-pgdg main" | sudo tee  /etc/apt/sources.list.d/pgdg.list
    $ sudo apt-get install postgresql-13 postgresql-client-13
    $ pg_basebackup -d "host=192.168.10.10 user=replicator
sslcert=$HOME/.postgresql/client-replicator.cert.pem
sslkey=$HOME/.postgresql/client-replicator.key.pem
sslrootcert=$HOME/.postgresql/ca-chain.cert.pem sslmode=verify-ca" -Fp -
Xs -P -R
    $ touch standby.signal
    $ systemctl restart postgresql.service && systemctl status
postgresql.service
```

4.4.4 Instalace webového serveru OpenResty a Node.JS

Pro provozování webových služeb je nutné nainstalovat a nakonfigurovat webový server. V této práci byl zvolen webový server OpenResty. Jedná se o verzi optimalizovanou na výkon serveru NginX, doplněnou o skriptovací jazyk Lua. Pro běh aplikace Aengine je nutné nainstalovat Node.JS. Tento software provádí interpretaci definice REST API zapsané pomocí OpenAPI 3. Oba druhy software se instalují ze svých repozitářů.

Instalace OpenResty je provedena posloupností následujících kroků.

```
    $ sudo systemctl disable nginx
    $ sudo systemctl stop nginx
    $ wget -O - https://openresty.org/package/pubkey.gpg | sudo apt-key
add -
    $ echo "deb http://openresty.org/package/arm64/ubuntu $(lsb_release
-sc) main" \
        | sudo tee /etc/apt/sources.list.d/openresty.list

    $ sudo apt-get update
    $ sudo apt-get -y install openresty
```

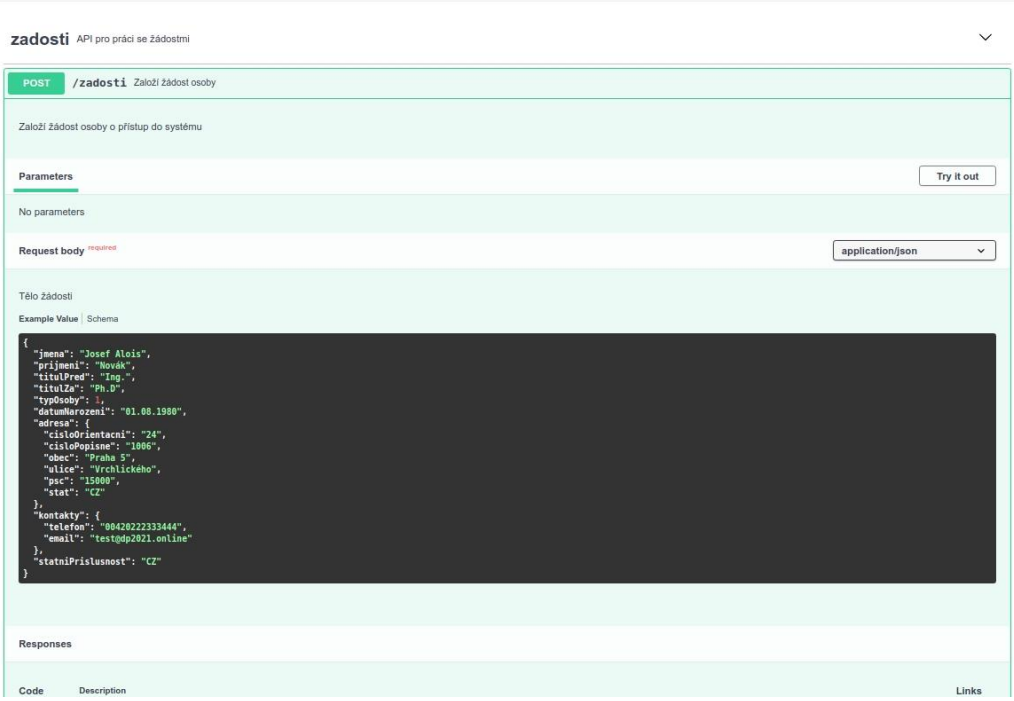
Instalace Node.JS se prování posloupností následujících kroků.

```
    $ curl -fsSL https://deb.nodesource.com/setup_14.x | sudo -E bash -
    $ sudo apt-get install -y nodejs
    $ sudo useradd -u 3000 -c "Node.JS ucet pro beh aplikaci" -g 3000 -d
"/opt/node" node
    $ sudo mkdir -p /var/lib/node/socket
    $ sudo chown node:nginx /var/lib/node/socket
    $ sudo chmod g+rxs /var/lib/node/socket/
```

4.4.5 REST API + OpenAPI 3

Definice REST API [47] vytvořená pro účely této práce je zapsána pomocí standardu OpenAPI 3. Definice obsahuje všechny CRUD operace včetně implementace. Na úrovni byznysové logiky se API koncipuje tak, že umožňuje založení jednoduché žádosti, její načtení, následnou změnu a zrušení.

Obrázek 8: Byznysová definice REST API



Zdroj: Vlastní zpracování

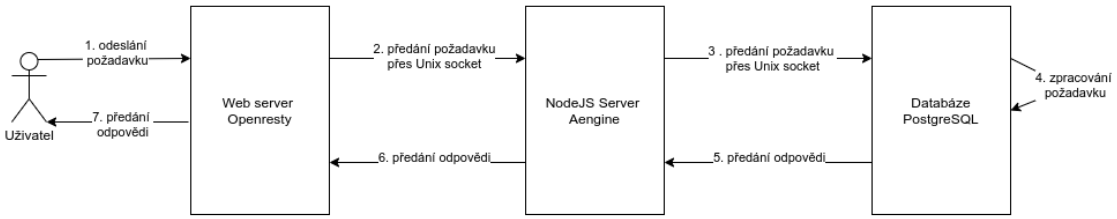
Implementace byznysové logiky se provede v backendu na úrovni funkcí v databázi PostgreSQL. Interpret definice OpenAPI Aengine je určený pro validaci vstupu, případně

výstupu a jeho klíčová funkce spočívá ve správném směrování jednotlivých http metod na zdroje v uložené v backendu.

Architektonicky je celá realizace zjednodušeně znázorněna následujícím diagramem, přičemž význam jednotlivých korků je následující.

1. Prostřednictvím protokolu HTTP je na OpenResty přijat požadavek. Je vyhodnoceno, zda se jedná o rouru */zadosti/v1* a pokud ano, dojde k předání do streamu node, tedy volání reverzní proxy na Unix socket, kde očekává požadavky Aegnine.
2. Aengine vyhodnotí na základě http metody a jejího názvu routu a dle hodnoty parametru operationId provede následující krok.
3. Je předáno řízení databázové funkci v PostgreSQL. Praktická realizace je uskutečněna volání příkazu SELECT na databázové funkci typu *security definer* prostřednictví nízko úrovňového protokolu *postgres* na Unix socketu.
4. Databáze PostgreSQL provede obsah funkce a vygeneruje odpověď
5. Odpověď je předání prostřednictvím protokolu *postgres* na Unix socketu
6. Aengine před odpověď pomocí protokolu HTTP na Unix socketu zpět na OpenResty
7. OpenResty předá výsledek volání REST API pomocí protokolu HTTP uživateli. V našem případě je uživatelem periodicky běžící program na Raspebrry Pi 400.

Obrázek 9: Tok zpráv v rámci implementace REST API



Zdroj: Vlastní zpracování

Definice OpeAPI 3 vytvořená jako součást vlastní práce a použitá v rámci testů je uložena v příloze této práce.

4.4.6 Konfigurace OpenResty a Aengie

Na úrovní Openresty se vytvoří pravidla, která definují obsluhu statického obsahu, tedy standardních webových stránek a jejich příslušenství a dále pak routy pro reverzní proxy. V níže uvedené konfiguraci je /zadosti/v1 definovaná pro reverzní proxy. Definice směrování je pak v rámci upstream, tedy pomocí Unix socketu.

```
    upstream dp2021 {
       server unix:/var/lib/node/socket/dp2021.socket fail_timeout=0;
      }

    location / {

            try_files $uri $uri/ @app;

            root   /usr/share/nginx/html/dp2021;
            index  index.html;
        }

    location = /zadosti/v1 {

            auth_ldap_servers lokalni_ldap;
            auth_ldap_servers replika_ldap;
            auth_ldap_require valid_user;

            proxy_set_header X-Forwarded-Host $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Remote-User $remote_user;
            proxy_set_header X-Real_IP $remote_addr;
            proxy_set_header Host $http_host;

            proxy_buffers 16 4k;
            proxy_buffer_size 2k;

            proxy_pass http://dp2021;

        }
```

V případě Aegine je konfigurace odlišná. Vytvoří se systemd skript, který Aengine spustí a umožní mu zpracovávat požadavky na Unix socketu. Celá konfigurace je pak zahrnuta v rámci tohoto skriptu a souboru /opt/node/dp2021, kde je uveden název souboru s definicí REST API pomocí specifikace OpenAPI 3.

```
    [Unit]
    Description=NodeJS API Diplomova prace
    After=syslog.target network-online.target
    Wants=network-online.target

    [Service]
    ExecStart=/usr/bin/node /opt/node/dist/server.js @/opt/node/dp2021 -
-path /opt/node/.env
    WorkingDirectory=/opt/node
    Restart=failure
    RestartSec=1s
    StandardOutput=syslog
    StandardError=syslog
    SyslogIdentifier=dp2021
    User=node
    Group=nginx
    Environment=NODE_ENV=production
    Environment=PORT="/var/lib/node/socket/dp2021.socket"
    Environment=NODE_PG_URL="socket:///var/run/postgresql?db=dp2021&user
=dp2021"
    Environment=LOG4JS_FILENAME=/opt/node/logs/dp2021.log

    [Install]
    WantedBy=multi-user.target
```

4.5 Zabezpečení clusteru

Cluster se zabezpečeje pomocí standardního firewallu [44] dodávaného v rámci distribuce Ubuntu. Jsou otevřeny pouze tyto porty:

- 80 a 443 pro komunikaci pomocí protokolu HTTP resp. jeho zabezpečenou verzí
- 25, 586 a 993 pro komunikaci protokoly SMTP a IMAPS pro emailový server
- 88, 749 a 4444 pro Kerberos
- 5432 pouze mezi replikami databáze PostgreSQL
- 389 a 636 pouze mezi replikami adresářového server OpenLDAP
- 9100 pro získávání data pro motitoring

4.5.1 Monitoring

Pro monitoring výkonu jednotlivých Raspberry Pi je využíván agent monitorovacího sytému Prometheus s názvem *node_exporter*. Monitoring je určen pro sběr dat týkajících se zatížení CPU, využití operační paměti, zaplnění a aktivity disků nebo aktivity síťových rozhraní. Data jsou v pravidelných intervalech odebírána z monitorovacího stroje mimo vlastní infrastrukturu a následně jsou zpracovávána pomocí databázových rozhraní.

Z bezpečnostních důvodů je monitoring spouštěn pod technickým uživatelem.

```
    $ sudo useradd -rs /bin/false node_exporter
```

Instalace se provede prostřednictvím následujícího předpisu. Spuštění je řízeno mechanismem systemd.

```
    $ curl –o node_exporter-1.1.2.tar.gz
https://github.com/prometheus/node_exporter/releases/download/v1.1.2/node
_exporter-1.1.2.linux-amd64.tar.gz
    $ tar -xf node_exporter-1.1.2.tar.gz
    $ sudo mv node_exporter-1.1.2.linux-amd64/node_exporter
/usr/local/bin
    $ sudo rm -r node_exporter-1.1.2.*
    $ sudo cat <<EOF > /etc/systemd/system/node_exporter.service
    [Unit]
    Description=Node Exporter
    After=network.target
    [Service]
    User=node_exporter
    Group=node_exporter
    Type=simple
```

```
ExecStart=/usr/local/bin/node_exporter
[Install]
WantedBy=multi-user.target
EOF
```

4.6 Online spolupráce

Možnost online spolupráce je jeden z nejdůležitějších atributů moderních řečení IT infrastruktury. V mnoha případech je spoluprací nazýváno i prosté sdílení souborů. V rámci této práce je představena možnost spolupráce na dokumentech i jiném obsahu v reálném čase, včetně možnosti správy verzí. Na platformě Raspberry Pi nejsou k dispozici rozsáhlé ekosystémy společnosti Microsoft. Bylo nutné realizovat spolupráci jednoduššími nástroji po stránce komplexnosti. Použité nástroje Gitea a EtherPad Lite je možné využít jako alternativu pro splnění značné část funckionality týkající se konfiguračního managementu, správy verzí a spolupráce s řádově nižšími požadavky na hardware.

4.6.1 Gitea

Nástroj Gitea [54] je nadstavba nad software pro správu verzí git. Na rozdíl od nástroje git, který je primárně určený pro vývojáře software a IT specialisty, Gitea je určena pro práci všem uživatelům prostřednictvím webového rozhraní a REST API. Gitea je plně integrována s výše popsaným ekosystémem, tedy s LDAP, OpenResty a PostgreSQL databází. Instalace nástroje Gitea je realizována dvoukrokově. Nejprve proběhne vlastní instalace software a následně se provede integrace nástroje do infrastruktury.

```
# Založení technického uživatele pro Gitea
$ sudo adduser --system --shell /bin/bash --group --disabled-
password --gecos 'Git Version Control' --home /home/git git

# Vytvoření uživatele a databáze gitea.
$ sudo su postgres
$ psql -c 'CREATE USER gitea WITH PASSWORD __HESLO__;'
```

```
$ psql –c 'CREATE DATABASE gitea OWNER gitea'
```

```
# Stažení poslední verze
$ export VER=1.13.6
$ sudo curl –o /usr/local/bin/gitea https://github.com/go- \
gitea/gitea/releases/download/v${VER}/gitea-${VER}-linux-arm64
$ sudo chmod +x /usr/local/bin/gitea

# Vytvoření adresářové struktury
$ sudo mkdir -p /etc/gitea \
/var/lib/gitea/{custom,data,indexers,public,log}
$ sudo chown git:git /var/lib/gitea/{data,indexers,log}
$ sudo chmod 750 /var/lib/gitea/{data,indexers,log}
$ sudo chown root:git /etc/gitea
$ sudo chmod 770 /etc/gitea

# Vytvoření systemd služby pro Gitea
$ cat <<EOF > /etc/systemd/system/gitea.service
[Unit]
Description=Gitea
After=syslog.target
After=network.target
After=postgresql.service

[Service]
RestartSec=2s
Type=simple
User=git
Group=git
WorkingDirectory=/var/lib/gitea/
ExecStart=/usr/local/bin/gitea web -c /etc/gitea/app.ini
Restart=always
Environment=USER=git HOME=/home/git GITEA_WORK_DIR=/var/lib/gitea

[Install]
WantedBy=multi-user.target
EOF
$ systemctl daemon-reload
```

Integrace Gitea na Raspberry Pi je specifická minimalizací ztrát výkonu změnou standardních nastavení. V nastavení se využije specifických funkcionalit operačního systému Linux a síťová komunikace se nahrazuje komunikací meziprocesovou. Aplikace Gitea se nastaví pouze pro naslouchání na Unix socketu, stejným mechanismem je realizováno připojení k databázi PostgreSQL Reverzní proxy Openresty se nastaví na předávání informací směrem do Gitea výhradně pomocí meziprocesové komunikace.

```
# Konfigurace streamu OpenResty pro /gitea
location /gitea/ {
    proxy_pass http://unix:/var/lib/gitea/gitea.socket:/;
}

# V konfiguraci Gitea je potřeba nastavit tyto hodnoty
[server]
PROTOCOL         = unix
SSH_DOMAIN       = dp2021.online
DOMAIN           = http://dp2021.online/gitea/
ROOT_URL         = http://dp2021.online/gitea/
HTTP_ADDR        = /opt/gitea/var/lib/gitea/gitea.socket
database]
DB_TYPE          = postgres
HOST             = /run/postgresql/
NAME             = gitea
USER             = gitea
PASSWD           = __HESLO__

# Vytvoření systemd služby pro Gitea
$ cat <<EOF > /etc/systemd/system/gitea.service
```

4.6.2 EtherPad Lite

Etherpad Lite [55] je online webový textový procesor, který podporuje spolupráci více uživatelů na dokumentech v reálném čase. Funkcionalitou je podobný online editoru dokumentů společnosti Google. Z pohledu bezpečnosti je zásadní skutečnost, že veškerá data jsou uložena v rámci konkrétní instance a řízení přístupu je podřízeno lokální přístupové politice. Aplikace EtherPad Lite je plně integrována s výše popsaným

ekosystémem, tedy s LDAP, OpenResty a PostgreSQL databází. Instalace nástroje Etherpad je realizována dvou krokovým mechanismem. Nejprve proběhne vlastní instalace software a následně je provedena integrace nástroje do infrastruktury.

```
    # Založení technického uživatele pro EtherPad
    $ sudo adduser --system --shell /bin/bash --group --disabled-
password --gecos 'EtherPad Lite' –home /home/etherpad etherpad

    # Vytvoření uživatele a databáze etherpad.
    $ psql –c 'CREATE USER etherpad WITH PASSWORD __HESLO__;'
    $ psql –c 'CREATE DATABASE etherpad OWNER etherpad'

    # Stažení a instalace poslední verze
    $ export VER=1.8.13
    $ sudo curl –o
/home/etherpad/etherpad.tar.gz https://github.com/ether/etherpad-
lite/archive/refs/tags/${VER}1.8.13.tar.gz
    $ sudo cd /home/etherpad && sudo tar -xzf ${VER}.tar.gz && sudo rm –
rf ${VER}.tar.gz && sudo chown –R etherpad:etherpad etherpad-lite-${VER}
&& sudo ln –s etherpad-lite-${VER} etherpad-lite && sudo chown
etherpad:etherpad etherpad

    # Vytvoření systemd služby pro Etherpad
    $ cat <<EOF > /etc/systemd/system/etherpad.service

    [Unit]
    Description=Etherpad Lite, the collaborative editor.
    After=syslog.target network.target
    Requires=etherpad.socket

    [Service]
    Type=simple
    User=etherpad
    Group=etherpad
    WorkingDirectory=/home/etherpad/etherpad
    ExecStart=/usr/bin/node /home/etherpad/etherpad/src/node/server.js
    StandardOutput=syslog
    StandardError=syslog
```

```
SyslogIdentifier=etherpad-lite
Restart=no

[Install]
WantedBy=multi-user.target
EOF
# Vytvoření socketu pro Etherpad
$ cat <<EOF > /etc/systemd/system/etherpad.socket
[Unit]
Description=Etherpad Lite socket for UNIX socket activation.
PartOf=etherpad.service

[Socket]
ListenStream=/tmp/etherpad.socket
Accept=False

[Install]
WantedBy=sockets.target

$ systemctl daemon-reload
```

Integrace pro přístup pomocí reverzní proxy a do databáze se realizuje pomocí stejného mechanismu jako v případě Gitea, tedy pomocí streamu do Unix socketu. Dále je možná hlubší integrace s LDAP pomocí modulu ep_ldapauth.

```
# Konfigurace streamu OpenResty pro /gitea
location /gitea/ {
    proxy_pass http://unix:/tmp/etherpad.socket:/;
}

# V konfiguraci Etherpad v souboru etherpad-lite/settings.json je potřeba nastavit tyto hodnoty

"dbType" : "postgres",
"dbSettings" : {
    "user"    : "etherpad",
    "host"    : "/var/run/postgresql",
    "password": "",
```

```
        "database": "etherpad",
        "charset" : "utf8mb4"
    }
```

5 Výsledky a diskuse

5.1 Naměřené a vypočtené hodnoty

V rámci práce bylo provedeno nejprve vyhodnocení rychlosti lokálního SSD disku. Metodika testů rychlosti SSD disku použitého v rámci Raspberry Pi byla následující. Spustily se testy čtení a zápisu, náhodného čtení a náhodného zápisu 4kB bloků a dvou základních testů propustnosti. To vše se opakovalo 10x a na závěr byla určena průměrná hodnota těchto testů spolu s odchylkou. Testy byly realizovány s využitím následujícího software:

- FIO – je průmyslový standard používaný k hodnocení skutečného výkonu úložiště v situacích reálného světa vs. pouhé měření propustnosti. Test prováděl sekvenční čtení a zápis a náhodné čtení a zápis 4kB bloků. Výsledek je v MB/s.
- DD Write - je test propustnosti zápisu co nejrychleji. To pomáhá určit maximální rychlost zápisu paměťového zařízení. Výsledek je v MB/s.
- HD Parm – je test propustnost přímého čtení disku bez ukládání do mezipaměti. Výsledek je v MB/s.

Dalším testem bylo měření propustnosti síťového karty integrované na desce. Raspberry Pi byly připojeny prostřednictvím UTP kabelu do switche TP Link TP-SG1008P, cože je osmiportový gigabitový switch s podporou Power ver Ethernet (PoE). Metodika testu rychlosti síťové kary podobně jako v případě testů rychlosti SDD disků byla postavena na použití níže uvedeného software, které bylo opakováno 10x.

- Rsync – je program na kopírování souborů. V rámci testu se kopíroval soubor velký 10 GB po síti s využitím programu rsync. To poskytlo dobrou představu o rychlosti přenosu dat při současném čtení, anebo zápisu soubor na SSD disk.
- Iperf – je program na měření propustnosti síťové karty. V rámci této práce je maximální teoretická propustnost 1000 Mb/s. V reálném světě by mělo být dosahováno hodnot více než 90% s teoretického maxima.

Dále bylo sledováno zatížení Raspbery PI v období od 4.1 2021 do 14.3 2021, tedy celkem 70 dní. Monitoring poskytoval data o zatížení prostřednictví parametru load average každou minutu. Zjednodušeně parametr hovoří o tom, kolik procesů bylo obslouženo bez čekání. Pokud tedy máme hodnotu např. 0,2 pak lze říci, že systém byl nečinný až z 80%. Za dané období bylo tedy změřeno více jak 100 tisíc hodnot. Tyto tři vazby byly použity k vytvoření intervalů pro všechny změřené hodnoty parametru load average.

Tabulka 2: Naměřené hodnoty výkonu síťové karty (RPi s SD karou)

Č. intervalu	**Rozsah load_average**	**Okamžitá spotřeba W**
1	<0; 0,08>	3,5
2	<0,09; 0,82>	5,5
3	<0,83; 3>	8

Zdroj: Vlastní zpracování

Posledním testem byla reálná simulace provozu pomocí volání REST API. Ve stejnou dobu jako probíhalo měření zatížení systému, byl realizování scénář dle následující metodiky:

- V rámci jedné jednotky testu byly volány za sebou v rámci jednoho záznamu čtyři REST API endpoint. Nejprve byl záznam založen metodou http metodou POST, následně byl načten http metodou GET, pak byl změněn http metodou PUT a nakonec zrušen pomocí metody DELETE. Volání všech operací proběhlo pomocí příkazu curl a provolán byl celý stack, tedy reverzní proxy, aplikační server i relační databáze.
- Volní bylo řízeno pomocí cronu.
- Každý den mezi 8 - 11 a 13 -16 hodinou byl test volán každou sekundu.
- Každý den mezi 11 – 13 byl test volán každé tři sekundy.
- Ve zbylém čase volání testu proběhlo 1x za 30 sekund.

Výsledný čas zpracování byl sbírán do logu reverzní proxy.

5.1.1 SSD disk vs. SD karta

V rámci testů byly naměřeny hodnoty uvedené v tabulce. Pro srovnání byly provedeny měření i s využitím SD karty.

Tabulka 3: Naměřené hodnoty výkonu SSD disku

Parametr	**Číslo měření**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
FIO KRw (kB/s)	81502	82604	81198	82365	82765	83234	82698	82669	83093	83123
FIO 4KRr (kB/s)	110156	105138	107234	110654	110958	109739	110467	108637	109232	109867
HDParm (MB/s)	321,3	318,9	319,1	319,1	319,8	319,2	319,6	320,0	319,6	319,8
DD Write (MB/s)	125	124	128	124	124	123	124	125	123	124

Zdroj: Vlastní zpracování

Tabulka 4: Naměřené hodnoty výkonu SD karty

Parametr	**Číslo měření**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
FIO KRw (kB/s)	4192	4223	4201	4208	4223	4176	4217	4187	4199	4206
FIO 4KRr (kB/s)	12314	12128	12156	12478	11967	12011	11865	12221	12398	12396
HDParm (MB/s)	41,85	40,67	41,56	41,50	41,62	40,99	40,91	41,21	41,84	40,37
DD Write (MB/s)	32	34	33	33	32	32	31	33	31	32

Zdroj: Vlastní zpracování

Tabulka 5: Vypočtené průměrné hodnoty výkonu SSD disku a SD karty

Parametr	Průměrná hodnota SSD disku	Průměrná hodnota SD karty
FIO KRw (kB/s)	82525,1 ± 642,7	4203,2 ± 14,7
FIO 4KRr (kB/s)	109208,2 ± 1707,1	12193,4 ± 194,2
HDParm (MB/s)	319,6 ± 0,7	41,3 ± 0,5
DD Write (MB/s)	124,4 ± 1,4	32,3 ± 0,9

Zdroj: Vlastní zpracování

Na základě změřených hodnot a průměrných hodnot získaných výpočtem, je možné konstatovat následující závěry.

- Použití SD kary zásadním způsobem limituje rychlost operací čtení a zápisu i za situace, kdy použije kvalitní a rychlou SD kartu. Rozdíly v počtu operací jsou řádové, pokud jde o výkon při čtení a zápisu pak je SSD disk násobně rychlejší oproti SD kartě. Je možné konstatovat, že poměr cena/kapacita/výkon je jednoznačně vychýlen směrem k použití SSD disku a pro podnikové použité je tedy SSD disk jednoznačnou volbou.
- Z pohledu technologie SSD disku je možné konstatovat, že díky architektuře Raspberry Pi nemá na výsledný výkon SSD disku vliv rozhraní SSD disku tedy technologie mSATA, M.2 SATA a M.2 NVMe. Raspberry Pi nedokáže využít maximální potenciál NVME SSD disků, jelikož limitujícím faktorem je propustnost USB 3.0 rozhraní, která je 5 Gbit/s, která je pro tuto architekturu tohoto jednodeskového počítače spíše teoretická.
- Použití SSD disku významným způsobem mění možné případy užití Raspberry Pi v reálném produkčním nasazení. Zatímco v případě použití SD karty je I/O nejslabším místem tohoto počítače, v případě užití SSD se nejslabším místem stává výkon procesoru. Tato změna umožňuje Raspberry Pi produkčně nasazovat na projekty z podnikového prostředí, který

nejsou náročné na výpočetní výkon (např. webové servery, infrastrukturní projekty jako např. LDAP nebo menší databáze primárně pracující v režimu OLTP). Navýšení výkonu lze řešit pomocí horizontálního škálování tedy pouhým přidáváním dalších Raspberry Pi.

5.1.2 Zhodnocení výkonu sítě

V rámci testů byly naměřeny hodnoty uvedené v tabulce. Pro srovnání byly provedeny měření i s využitím SD karty.

Tabulka 6: Naměřené hodnoty výkonu síťové karty (RPi s SSD)

Parametr	**Číslo měření**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
LAN->Pi (MB/s)	75,2	76,1	75,6	74,9	75,3	75,4	75,4	75,9	76,0	75,9
Pi ->LAN (MB/s)	94,1	95,3	95,9	95,2	94,8	94,9	95,3	94,3	94,6	95,7
iperf Mbps	933	929	934	938	937	933	929	933	931	933

Zdroj: Vlastní zpracování

Tabulka 7: Naměřené hodnoty výkonu síťové karty (RPi s SD karou)

Parametr	**Číslo měření**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
LAN->Pi (MB/s)	4,2	4,2	4,3	4,1	4,0	4,3	4,0	3,9	4,2	4,1
Pi ->LAN (MB/s)	11,8	11,5	11,7	11,8	11,3	11,9	11,0	11,3	11,4	11,8
iperf Mbps	936	931	933	929	930	932	929	929	933	929

Zdroj: Vlastní zpracování

Tabulka 8: Vypočtené průměrné hodnoty výkonu síťové karty při použití SSD disku a SD karty

Parametr	**Průměrná hodnota síť na SSD disku**	**Průměrná hodnota síť na SD kartě**
LAN->Pi (MB/s)	77,6 ± 0,4	4,1 ± 0,1
Pi->LAN (MB/s)	95,0 ± 0,6	11,6 ± 0,3
iperf Mbps	933 ± 2,8	931 ± 2,3

Zdroj: Vlastní zpracování

Na základě změřených hodnot a průměrných hodnot získaných výpočtem, je možné konstatovat následující závěry.

- Použití SD karty limituje rychlost síťových operací, pokud současně probíhá čtení nebo zápisu s využitím tohoto zařízení. Výkon síťové karty pak odpovídá rychlosti, kterou je schopna data poskytovat nebo zapisovat SD karta. Pokud neprobíhají v rámci síťové komunikace žádné operace se soubory, pak je výkon síťové karty konstantní.
- Pro reálné podnikové užití je výkon síťové karty dostačující. Omezení výkonu je taká patrné, pokud se operací účastní procesor, např. pokud v rámci Raspberry Pi provozujeme VPN. Výslednou propustnost sítě pak limituje právě výkon procesoru.
- Pro operace typu poskytování statického obsahu skrze webový server se výkon sítě bez užití SD karty, vyrovná běžným hostingovým službám.

5.1.3 Energetická náročnost

Na základě naměřených dat o zátěži, které byly k dispozici každou minutu, proběhlo následující zpracování:

- Každý údaj o zátěži obdržel zařazení podle tabulky č. X. V dalším kroku se pracovalo již pouze s údaji o zařazení.
- V rámci každé hodiny byla provedena agregace naměřených hodnot a určeny nejčastěji se vyskytující údaje o zařazení. Výpočet byl proveden pro všech 70 dní, kdy probíhalo měření. V dalším kroku zpracování se používaly pouze agregované údaje po hodině.
- Následně byl určen pomocí agregace výskyt nejčastější hodnoty v rámci každé hodiny během dne přes rámec 70 dní. Tak vznikla tabulka určující nejčastější hodnoty spotřeby v rámci dne a bylo s ní možné provést finální výpočet spotřeby za rok.

Tabulka 9: Spotřeba energie při běžném provozu

Hodina	Interval spotřeby	Spotřeba (h) kWh	Spotřeba (rok) kWh	Roční cena Kč
0	1	0,0035	1,2775	6,40
1	1	0,0035	1,2775	6,40
2	1	0,0035	1,2775	6,40
3	1	0,0035	1,2775	6,40
4	1	0,0035	1,2775	6,40
5	1	0,0035	1,2775	6,40
6	1	0,0035	1,2775	6,40
7	1	0,0035	1,2775	6,40
8	3	0,008	2,92	14,63
9	3	0,008	2,92	14,63
10	3	0,008	2,92	14,63
11	2	0,0055	2,0075	10,06
12	2	0,0055	2,0075	10,06
13	3	0,008	2,92	14,63

14	3	0,008	2,92	14,63
15	3	0,008	2,92	14,63
16	2	0,0055	2,0075	10,06
17	2	0,0055	2,0075	10,06
18	2	0,0055	2,0075	10,06
19	1	0,0035	1,2775	6,40
20	1	0,0035	1,2775	6,40
21	1	0,0035	1,2775	6,40
22	1	0,0035	1,2775	6,40
23	1	0,0035	1,2775	6,40
Celkem		**0,121**	**44,165**	**221,28**

Zdroj: Vlastní zpracování

Cena byla určena na základě cenové sazby 5,01 Kč za kWh uvedené na vyúčtování. Za tuto sazbu dodává energii Pražská energetika a.s. malé nevýrobní firmě. Vzhledem k tomu, že období, které je k dispozici je dostatečně reprezentativní, lze poměrně přesně odhadnout cenu za energie a rok pro kalkulace do období jednoho roku a 3 let. Na základě těchto údajů je možné vyslovit závěr, že Raspberry Pi je energeticky velmi úsporné zařízení i za předpokladu intenzivního využívání.

5.1.4 Časy odezvy na REST API

Na základě naměřených dat o době odezvy, které byly k dispozici pro každé volání, proběhlo následující zpracování:

- Byly agregovány hodnot délky jednotlivých http volání v rámci každé hodiny během dne přes rámec 70 dní.
- Z těchto hodnot byl určen její 95% percentil. Tak vznikla tabulka určující maximální čas odezvy 95% konkrétního typu http volání v rámci konkrétní hodiny během dne.
- Uvedené hodnoty v tabulce jsou v sekundách

Tabulka 10: 95% percentil pro jednotlivé typy http volání v rámci dne

Hodina	**http metoda REST volání (sekundy)**			
	POST	**GET**	**PUT**	**DELETE**
0	0,128	0,061	0,134	0,098
1	0,124	0,062	0,135	0,096
2	0,126	0,062	0,134	0,095
3	0,129	0,061	0,135	0,097
4	0,128	0,061	0,134	0,095
5	0,126	0,061	0,136	0,097
6	0,128	0,062	0,135	0,098
7	0,129	0,061	0,134	0,098
8	0,142	0,068	0,158	0,109
9	0,145	0,068	0,154	0,110
10	0,145	0,067	0,155	0,109
11	0,138	0,066	0,144	0,104
12	0,138	0,066	0,144	0,107
13	0,147	0,067	0,156	0,109
14	0,145	0,068	0,158	0,110
15	0,146	0,068	0,155	0,111
16	0,138	0,067	0,146	0,108

17	0,138	0,066	0,144	0,106
18	0,135	0,065	0,154	0,103
19	0,129	0,063	0,138	0,102
20	0,126	0,063	0,137	0,099
21	0,128	0,062	0,134	0,100
22	0,128	0,062	0,135	0,099
23	0,127	0,063	0,134	0,098

Zdroj: Vlastní zpracování

Z tabulky určíme nejvyšší hodnoty http metod POST, GET, PUT a DELETE.

Tabulka 11: nejvyšší hodnoty 95% percentil pro jednotlivé typy http volání

http metoda REST volání (sekundy)			
POST	**GET**	**PUT**	**DELETE**
0,147	0,068	0,158	0,111

Zdroj: Vlastní zpracování

5.2 Zhodnocení výkonu Rasperry Pi

Je možné konstatovat, že čtecí operace jsou rychlejší než operace zápisu. Raspberry Pi v konfigurace popsané výše dokáže obsloužit bez zpoždění, které by bylo způsobené čekáním na přidělení zdrojů až čtyři požadavky na REST API rozhraní sekundu. Zde je třeba zdůraznit, velice důležité okolnosti, které umožnili dosáhnout těchto výsledků:

- Záleží na volbě konkrétního software, zejména je důležitá podpora některých specifických vlastností operačního systému Linux
- Architektura založená na událostech je důležitá pro maximální propustnost na zařízení, které není předimenzováno po stránce výkonu.
- Je nutné maximálně využít možností, které poskytuje konkrétní operační systém a minimalizovat zbytečné prodlevy způsobené použitím nejjednodušší cesty. Typickým příklad je Unix socket oproti síťovému volání na localhostu.

V rámci testů bylo spuštěno REST API i na platformě x86, konkrétně ODROID-H2+ x86. Jednodeskový počítač měl podobné parametry tedy 4 jádrový procesor Intel Quad-core J4115, 8 GB RAM, běžel na NVMe SSD disku a byl připojený do gigabitové sítě.

Tabulka 12: nejvyšší hodnoty 95% percentil pro jednotlivé typy http volání

Architektura	**http metoda REST volání (sekundy)**			
	POST	**GET**	**PUT**	**DELETE**
Raspberry Pi 4 ARM	0,147	0,068	0,158	0,111
ODROID-H2+ x86	0,117	0,050	0,125	0,080

Zdroj: Vlastní zpracování

Z výsledků je patrné, že výkon na platformě ODROID-H2+ byl o cca. 25% vyšší než na Raspberry Pi 4. Z naměřených hodnot je možné učinit následující závěry:

- Doba udezvy není úměrná výkonu jednodeskového počítače. Je třeba rozlišitovat mezi dobou odezvy REST API a spotřebovaným výkonem. Doba

odezvy souvisí s frekvencí procesoru a rychlostí provádění jednotlivých operací. Výkon je pak dán celkovou architekturou a použitými komponentami.

- Raspberry Pi 4 se blížilo ve špičce na hranici 80% spotřeby výkonu, zatímco platforma ODROID-H2+ byla na polovině této hodnoty a bylo na ní možné se stejnou dobou odpovědi provozovat dvojnásobné množství požadavků. Na Raspberry Pi 4 by snaha o zvýšení výkonu vedla k zahlcení a následně by se doba odpovědí začala prodlužovat.
- Platforma Raspberry Pi 4 vykazovala v průměru 3x nižší spotřebu energie oproti ODROID-H2+ při stejné činnosti.
- Pořizovací náklady u platformy Raspberry Pi 4 poloviční oproti platformě ODROID-H2+.

Je možné prohlásit, že Raspberry Pi 4 je schopno fungovat jako levná alternativa v rámci infrastruktury malé firmy, pokud je firma si vědoma podmínek provozu a případných omezení této platformy po stránce výkonu. Je proto nutné posuzovat konkrétní případy užití vždy komplexně v kontextu očekávání uživatelů. Testovaný scénář, kdy REST API obsloužilo 29 160 pořadavků za den a ve špičce obsluhovalo 4 požadavky za sekundu je poměrně hodně nadsazený, oproti běžnému provozu v mnoha firmách. Spíše než o vhodnosti Raspberry Pi v kontextu velikosti firmy je vhodné vést diskuzi o případech užití.

Současně je nutné konstatovat, že zcela klíčová pro dosažení výše popsaných hodnot je dobře navržená architektura, která plně reflektuje možnosti opračního systému, moderních technologií a požadovaného případu užití. Nevhodné užití software nebo špatná implementace dokáže degradovat prakticky jakýkoliv výkon serverového řešení.

5.3 Zhodnocení režimu vysoké dostupnosti

V rámci práce bylo provedeno otestování tří různých softwarových technologií z pohledu režimu vysoké dostupnosti. Tyto technologie jsou velmi často využívané v rámci softwarové infrastruktury v podnikovém prostředí:

- LDAP cluster – je téměř výhradně používán pro autentizaci, případně autorizaci uživatelů. V běžném podnikovém prostředí jsou jiné servery pro vnitřní užití, především pokud se jedná o sítě s operačním systémem Windows, kde se využívá Active Directory. Pro autentizaci externích uživatelů pak bývá provozován samostatný LDAP cluster s využitím svobodného software především z licenčních důvodů.
- Databázový cluster PostgreSQL – je stabilní objektově relační databáze, která v podnikovém prostředí doplňuje nebo stále častěji nahrazuje komerční databáze Oracle nebo Microsoft SQL. Tím, že je PostgreSQL možné horizontálně škálovat a podporuje provoz v režimu vysoké dostupnosti, je možné na tuto technologii nasadit i kritické podnikové aplikace.
- Cluster webového serveru a reverzní proxy – je další pokročilý případ užití, kde se kombinuje síla webového serveru, reverzí proxy a aplikačního serveru na bázi Node.JS. Tento cluster lze škálovat pouhým startováním dalších instancí.

Způsob řešení výpadku u jednotlivých clusterů je řešen v samostatných podkapitolách.

5.3.1 Zhodnocení vysoké dostupnosti LDAP

V předchozích kapitolách bylo specifikováno, jak lze nastavit LDAP cluster do režimu vysoké dostupnosti prostřednictvím replik. Klienti LDAP se nastaví tak, aby přistupovali více jak jednu LDAP repliku (minimálně dvě, ideálně tři). To znamená, že výpadek jakékoliv repliky nezpůsobí přerušení poskytování služby. Nefunkční replika se nahrazuje pouhým vystartováním nové repliky v případě multimaster replikace. V případě jednoduché master/slave replikace se v případě havárie masteru provede obnova ze zálohy v případě slave se opět vystartuje další replika. Proto je od začátku vhodné v případě LDAP uvažovat o multi master replikaci.

Na Rasperry PI lze bez problémů provozovat cluster s multimaster replikací. Konfigurace je naznačena v praktické části. V rámci vyhodnocení použitelnosti se služby na REST API autentizovali proti LDAP clusteru. LDAP cluster bez problémů odpovídali na více než deset požadavků za sekundu za situace, kdy byla jedna replika fyzicky odpojena od sítě pro simulaci výpadku.

5.3.2 Zhodnocení vysoké dostupnosti objektově relační databáze PostgreSQL

Nejjednodušším způsobem řešení vysoké dostupnosti je matser/slave replikace. Replikace může probíhat v asynchronním a synchronním režimu. Při asynchronním režimu transakce při commit na masteru nečeká na provedení commit i v replice, což znamená úsporu času při provádění transakce. Cenou za úsporu času je teoretická možnost nekonzistence mezi master a slave při pádu master instance databáze při velké zátěži. Je potřeba vyhodnotit reálnou možnost takového stavu a jeho dopady. Extrémními případy pro srovnání je bankovní systém pro vypořádání finančních transakcí a systém pro webovou prezentaci. V prvním případě potřebujete v případě havárii přepnout na repliku a mít jistotu, že replika 100% odpovídá originálu těsně před pádem a nelze akceptovat žádnou chybu. V druhém případě není případné ztráta jedné transakce akceptovatelné riziko. Případný uživatel, kterého by se tato operace dotkla, danou transakci zopakuje znova.

Jak bylo zmíněno dříve, PostgreSQL podporuje tzv. fyzickou replikaci (tedy replikaci po blocích nad celou databázovou instancí) a to jak v režimu master/slave tak i jako kaskádovou replikaci. Toto je nejjednodušší způsob řešení vysoké dostupnosti. V případě výpadku jakékoliv slave instance se tato nahradí pouhým vystartováním nové bez dopadu na běh systému. V případě, že dojde k výpadku master instance, postupuje se tak, že se libovolná slave instance přepne do režimu master instance a tím je umožněn zápis. Pokud se nastaví tato master instance na stejné IP jako původní, všechny slave instance naváží na replikaci automaticky z nové master instance. Tato operace je nevratná.

Pokročilejším režimem je tzv. sharding nebo využití logické replikace. V případě shardingu jde o rozložení jedné logické databáze do více databázových instancí PostgreSQL. Zápis do různých instancí pak probíhá podle klíče, kterým může být např. ID zákazníka. Tím se minimalizuje riziko dopadu případného výpadku na omezenou skupinu uživatelů a významně se zvyšuje dostupnost. Logická replikace probíhá na úrovni

jednotlivých tabulek a je na rozdíl od fyzické replikace možná i v rámci různých verzí PostgreSQL. Tímto mechanismem je možné realizovat i systémy, které vyžadují velmi vysoký tok na zápis, např. pro API na sběr dat. Zápis pak probíhá do více clusterů paralelně pro rozložení výkonu a pomocí logické replikace jsou pak data slučována do jedné cílové databáze.

V rámci testů byly provedeny simulaci výpadků postupným odepínáním master a slave instance. V případě výpadku slave instance nebyl pozorován reálný dopad na aplikaci ani v případě nejvyšší zátěže. Pro případ výpadku master instance dojde k několikavteřinovému výpadku během přepínání slave instance na master. Než dojde k přepnutí, nelze do zbylých instancí databází zapisovat, tedy služba nefunguje správně pro uživatele využívající zápis. Také toto přepnutí bylo otestováno během největší zátěže s výsldkem, kdy odezva několik jednotek požadavků skončila s chybou. Bylo vyhodnoceno, že provést zcela automatické přepnutí při jakémkoliv výpadku (např. několikavteřinové nedostupnosti sítě) je kontraproduktivní, jelikož se jedná o nevratnou operaci. Výhodnější je databázi monitorovat a provést manuální zásah na základě vyhodnocení celého incidentu. Případ chvilkového výpadku sítě bez přepnutí požadavky obsloužil bez chyb, pouze se zpoýdením.

5.3.3 Zhodnocení vysoké dostupnosti clusteru webserveru a reverzní proxy

V tomto konkrétním případě za atomickou jednotku považujeme instanci, která zahrnuje jak webový server, tak případnou aplikaci běžící např. v NodeJS, ke které webový server poskytuje reverzní proxy. Jako příklad budiž tedy jedna instance testovacího API pro účely této práce.

Vysoká dostupnost je řešena na úrovni DNS záznamu jako tzv. DNS Round-Robin. V DNS se A záznamy nastaví tak, že všechny IP adresy jednotlivých instancí odkazují na stejné doménové jméno. Jednotlivé DNS servery pak vrací permutovaný seznam IP adres. V případě, že o překladu adres rozhoduje lokální DNS server, dochází k situaci, že každý požadavek obsluhuje jiná IP a dochází k rotaci. V případě, že dojde k výpadku, je záznam konkrétní instance vyřazen z DNS. Prakticky bylo ověřeno odpojením jedné instance, že tato technika funguje spolehlivě, bez přerušení poskytování služby. Stejně tak po obnovení spojení a DNS záznamu se instance opět zapojila do poskytování služby.

Pokročilejším způsobem, který nevyžaduje změny v DNS záznamech je řešení s využitím HA Proxy. Předpokladem pro tento typ řešení je nutnost tento software nainstalovat na router nebo firewall. Dále jak pak každou instanci clusteru nutné doplnit o routu, která poskytuje informaci o stavu. HA proxy aktivně zjišťuje dostupnost jednotlivých instancí v clusteru a pokud nějaké přestane odpovídat, je automaticky vyřazena.

Každý ze způsobů má své výhody i nevýhody a rozhodnutí o volbě konkrétního způsobu je závislá jak na implementaci, možnostech konkrétní společnosti, ale i na poskytované službě a požadovaných parametrech dostupnosti.

5.4 Srovnání s Raspberry Pi s alternativami

Pro srovnání máme firmu, které má 9 zaměstnanců. Varianty budou Raspberry 4B+ v ceně 4 489,- Kč za kus, dále pak Public Cloud b2-7 a c2-7 u velkého poskytovatele v Evropě tedy jako IAAS zdroj[]. Srovnání s GSuite od Google je uvedeno pro ilustraci [].

Přestože by mohlo být namítnuto, že Google nabízí kromě Cloudu i aplikace, je na místě připomenout, že i řešení s Rasperry PI nabízí vše v on permise variantě v podobně opensource aplikací jako součást Linuxové distribuce. Naproti tomu public Cloud něco takového nenabízí, zde se jedná čistě o infrastrukturu. I tak je možné obě varianty s Raspberry Pi porovnat, protože jde o dosažení podobných výsledků s různou infrastrukturou.

Celé řešení je možné realizovat jako alternativu i ke stávajícím řešením v rámci firmy, jelikož na trhu je stále dost společností, které řeší infrastrukturu lokálním serverem. Pro reálné srovnání je pak potřeba vzít konkrétní pořizovací náklady, počet serverů, cenu za podporu a SLA, náklady na energie a porovnat to s náklady na pořízení a provoz Raspberry Pi.

5.4.1 Cenová kalkulace řešení v různých časových horizontech

Celkovou cenu provozované infrastrukturu na Raspberry Pi lze stanovit jako součet fixních nákladů na pořízení hardware a variabilních nákladů zahrnující cenu energie. Hardware jsme dimenzovali tak, že řešení je v režimu 2 + 1. Tedy poskytované IT služby běží na dvou počítačích Raspberry Pi 4B+ a třetí počítač Raspberry Pi 4+ plus je nakonfigurovaný tak, aby mohl okamžitě nahradit stroj, který hardwarově selhal a poskytovat služby během reklamace.

Cenová kalkulace pro 3 ks Raspberry Pi 4B+ včetně NVME SSD disků jsou v následující tabulce. Za uvedenou cenu byly zařízení i s moduly nakoupeny společností, která je k provedení testů poskytla. Jedná se o jednorázové, tedy fixní náklady. Cenová kalkulace pro elektrickou energii vychází z běžné ceny elektrické energie dodavatele Pražská energetika a.s. v sazbách, které jsou poskytovány podnikatelům a nevýrobním

firmám kategorie SME. Jedná se tedy o variabilní náklady. Je zřejmé, že cena energie je výrazně nižší než cena za pořízení vlastního zařízení, proto lze zanedbat i případné chyby, které vznikly aproximací spotřeby na delší období.

Cenový kalkulace Public cloudu vychází z paušální platby za měsíc [52], předpokládáme, že stejně jako Raspberry Pi cloud funguje 24x7. Za rok resp. tři roky se jedná o 12 resp. 36 paušálních měsíčních plateb.

Cenová kalkulace GSuite od Google vychází z ceny za jednoho uživatele a měsíc [53]. Tedy cena za období 12 resp. 36 měsíců vychází z měsíční platby za 9 uživatelů, což je skupina, která služby clusteru využívala.

Ceny jsou v korunách bez DPH. Pro výpočet byl použit kurz 26,80 Kč za EUR.

Tabulka 13: Vyčíslení nákladů za 1 rok

Varianta	Fixní náklady (Kč)	Variabilní náklady (Kč)	Celkem (Kč)
3x Raspberry Pi	13 467	221,21	13 688,21
Public Cloud - IAAS	0	16 723,20	16 723,20
GSuite - SAAS	0	29 764,80	29 764,80

Zdroj: Vlastní zpracování

Tabulka 14: Vyčíslení nákladů za 3 roky

Varianta	Fixní náklady (Kč)	Variabilní náklady (Kč)	Celkem (Kč)
3x Raspberry Pi	13 467	221,21	14 130,63
Public Cloud - IAAS	0	50 169,60	50 169,60
GSuite - SAAS	0	89 294,40	89 294,40

Zdroj: Vlastní zpracování

Z výsledků je zřejmé, že v delším časovém horizontu se vyplatí řešení na vlastním hardware. A to i za cenu, že zaplatíme osobu, která řešení provozně nastaví.

6 Závěr

V rámci diplomové práce byla realizována IT infrastruktura na bázi počítače Raspberry Pi 4B+ a konfigurace byla převedena do opakovatelné formy. Následně bylo provedeno v rámci pilotního produkčního nasazení měření zátěže a výpočet spotřeby takového řešení. Z těchto údajů byla dále sestavena cenová kalkulace pro období jednoho roku a tří let. Dále bylo provedeno měření odezvy reálného REST API na Raspberry Pi 4B a srovnání s provozem na architektuře x86.

Raspberry Pi 4B+ může sloužit jako základ kvalitní IT infrastruktury podnikatele nebo malé firmy, pokud firma provozuje moderní aplikace provozovatelé na operačním systému Linux. Důležité je využít maximálně potenciál zařízení pomocí optimalizace na operační systém Linux v kombinaci s moderní architekturou software a zařízením nabízející rychlé I/O operace. Na Raspberry Pi lze provozovat stabilně i relační databáze ve vysoké dostupnosti a horizontálně škálovat výkon přidáváním dalších jednotek. To bylo potvrzeno výsledky práce.

Za minimálních pořizovací náklady, lze provozovat několik let IT infrastrukturu, které bude udržitelná a současně bude mít zcela minimalistické energetické nároky. Vzhledem k tomu, že tento parametr je v rámci EU velmi aktuální, pro mnoho firem může být i toto způsob, jak dosáhnout nižších emisí a uspořit náklady. Celé řešení je provozovatelné i s použitím alternativních energetických zdrojů.

Práce prezentovala některé možnosti užití Rasperry PI 4B jako infrastruktury, nicméně výčet dalších případů užití je mnohem bohatší. Lze konstatovat, že pokud bude řešení dále vyvíjeno a rozšířeno o další moduly může sloužit jako velmi zajímavá distribuovaná IT infrastruktura nejenom v malých firmách, ale prakticky ve všech menších a středních projektech bez ohledu na velikost firmy. Nízké pořizovací náklady dělají z Raspberry Pi 4B+ velmi zajímavou alternativu k datovým centrům. Je možné je za velmi příznivých podmínek distribuovat do různých geografických lokalit a dále snížit náklady na připojení a vytvářet distribuované, robustní a bezpečné firemní infrastruktury.

7 Seznam použitých zdrojů

[1] OECD Glossary of Statistical Terms. *Small and medium-sized enterprises* [online]. [cit. 2021-03-29]. Dostupné z: https://stats.oecd.org/glossary/detail.asp?ID=3123

[2] Vláda ČR. *Usnesení č. 431/2020 Sb.* [online]. [cit. 2021-03-29]. Dostupné z: https://www.zakonyprolidi.cz/cs/2020-431

[3] Raspberry Pi Foundation. *Raspberry Pi 4 Model B product brief* [online]. (PDF). [cit. 2021-03-29]. Dostupné z: https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-product-brief.pdf

[4] Raspberry Pi Foundation. *Raspberry Pi Compute Module 4 IO Board product brief* [online]. (PDF). [cit. 2021-03-29]. Dostupné z: https://datasheets.raspberrypi.org/cm4io/cm4io-product-brief.pdf

[5] Suptronics. *Hardware tutorial* [online]. [cit. 2021-03-29]. Dostupné z: http://www.suptronics.com/miniPCkits/x872-hardware.html

[6] Raspberry Pi Foundation. *Raspberry Pi 4 400 product brief* [online]. (PDF). [cit. 2021-03-29]. Dostupné z: https://datasheets.raspberrypi.org/pi400/pi400-product-brief.pdf

[7] Ubuntu Downloads. *Install Ubuntu on a Raspberry Pi* [online]. [cit. 2021-03-29]. Dostupné z: https://ubuntu.com/download/raspberry-pi

[8] Raspberry Pi Foundation. *Raspberry Pi OS* [online]. [cit. 2021-03-29]. Dostupné z: https://www.raspberrypi.org/software/

[9] Ubuntu Blog. *Ubuntu 20.04 LTS is certified for the Raspberry Pi* [online]. [cit. 2021-03-29]. Dostupné z: https://ubuntu.com/blog/ubuntu-20-04-lts-is-certified-for-the-raspberry-pi

[10] Ubuntu About. *The Ubuntu lifecycle and release cadence* [online]. [cit. 2021-03-29]. Dostupné z: https://ubuntu.com/about/release-cycle

[11] Red Hat, Inc. *Red Hat Enterprise Linux Life Cycle* [online]. [cit. 2021-03-29]. https://access.redhat.com/support/policy/updates/errata

[12] Red Hat, Inc . *CentOS Stream: Building an innovative future for enterprise Linux* [online]. [cit. 2021-03-29]. Dostupné z: https://www.redhat.com/en/blog/centos-stream-building-innovative-future-enterprise-linux

[13] Rocky Linux. *A community-driven effort to bring you enterprise-grade, production-ready Linux.* [online]. [cit. 2021-03-29]. Dostupné z: https://rockylinux.org/

[14] Alma Linux. *An open-source RHEL fork, formerly known as Project Lenix*. [online]. [cit. 2021-03-29]. Dostupné z: https://almalinux.org/

[15] CentOS Blog . *CentOS Project shifts focus to CentOS Stream* [online]. [cit. 2021-03-29]. Dostupné z: https://blog.centos.org/2020/12/future-is-centos-stream/

[16] MATOTEK, Dennis, TURNBULL James, LIEVERDINK Peter. *Pro Linux System Administration, 2nd Edition.* 2017 Apress; ISBN 1484220072

[17] JEFTOVIC, Mark E. *Managing Mission-Critical Domains and DNS*. Packt Publishing, 2018. ISBN 1789135079

[18] SCHÖNIG, Hans-Jürgen. *Mastering PostgreSQL 13, 4th Edition.* Packt Publishing, 2020. ISBN 1800567499

[19] PIROZZI, Enrico, FERRARI, Luca. *Learn PostgreSQL.* Packt Publishing, 2020. ISBN 978-1838985288

[20] SHAUN, Thomas. *PostgreSQL 12 High Availability Cookbook, Third Edition*. Packt Publishing, 2020. ISBN 978-1838984854

[21] KUMAR, Vallarapu Naga Avinash. *PostgreSQL 13 Cookbook.* 2021 Packt Publishing; ISBN 1838648135

[22] SCHÖNIG, Hans-Jürgen. *PostgreSQL Replication, 2nd Edition*. 2015 Packt Publishing; ISBN 1783550600

[23] VAZQUEZ, Antonio. *Practical LPIC-3 300: Prepare for the Highest Level Professional Linux Certification.* Apress, 2019. ISBN 1484244729

[24] HELMKE, Matthew. *Ubuntu Linux Unleashed 2021 Edition, 14th Edition*. Addison-Wesley Professional, 2020. ISBN 0136778852

[25] MCNAB, Chris. *Network Security Assessment: Know Your Network, 3rd Edition.* O'Reilly Media, 2016. ISBN 978-1491910955

[26] DOSTÁLEK, Libor, KABELOVÁ, Alena. *Velký průvodce protokoly TCP/IP a systém DNS.* Computer Press, 2008. ISBN 978-80-251-2236-5

[27] DOSTÁLEK, Libor. *Kerberos včera, dnes a zítra*. [online]. (PDF). [cit. 2021-03-29]. Dostupné z: https://www.cleverandsmart.cz/wp-content/uploads/Kerberos-vcera-dnes-a-zitra.pdf

[28] DOSTÁLEK, Libor. *Mobilní sítě a jejich bezpečnost.* Ústav aplikované informatiky, Přírodovědecká fakulta Jihočeské univerzity v Českých Budějovicích, 2016. ISBN 978-80-7394-606-7

[29] NEGUS, Christopher. *Linux Bible, 10th Edition.* Wiley, 2020 ISBN 1119578884

[30] SONI, Rahul. *Nginx: From Beginner to Pro.* Apress, 2016; ISBN 1484216571

[31] AIVALIOTIS, Dimitri. *Mastering NGINX, 2nd Edition.* Packt Publishing, 2016. ISBN B01BP23ILY

[32] NEDELCU, Clément. *Nginx HTTP Server, 3rd Edition.* Packt Publishing, 2015. ISBN 1785280333

[33] ZHANG, Yichun. *agentzh's Nginx Tutorials.* [online]. [cit. 2021-03-29]. Dostupné z: https://openresty.org/download/agentzh-nginx-tutorials-en.html

[34] MAMMINO, Luciano, CASCIARO, Mario. *Node.js Design Patterns: Design and implement production-grade Node.js applications using proven patterns and techniques, 3rd Edition.* Packt Publishing, 2020. ISBN 1839214112

[35] MASSE, Mark. *REST API Design Rulebook 1st Edition.* O'Reilly Media, 2011. ISBN 1-4493-1050-8.

[36] RITCHIE, Dennis, *THOMPSON, Ken. "The UNIX time-sharing system"* [online]. (PDF). [cit. 2021-03-29]. Dostupné z: https://people.eecs.berkeley.edu/~brewer/cs262/unix.pdf

[37] RAYMOND, Eric S. *"Basics of the Unix Philosophy". The Art of Unix Programming.* Addison-Wesley Professional, 20994. ISBN 0-13-142901-9

[38] RedHat. redhat.com. [online]. [cit. 2021-03-29]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api.

[39] DOGLIO, Fernando. *REST API Development with Node.js: Manage and Understand the Full Capabilities of Successful REST Development 2nd ed. Edition*. Apress, 2018. ISBN 978-1484237144.

[40] Swagger. swagger.io. [online]. [cit. 2021-03-29]. Dostupné z: https://swagger.io/docs/specification.

[41] OpenAPI Initiative. openapis.org. [online]. [cit. 2021-03-29]. Dostupné z: http://spec.openapis.org/oas/v3.0.3.

[42] LAURET, Arnaud. The Design of Web APIs 1st Edition. Manning Publication, 2019. ISBN 978-1617295102.

[43] JSON json.org. [online]. [cit. 2021-03-29]. Dostupné z: https://www.json.org/json-en.html.

[44] SUEHRING, Steve. *Linux Firewalls: Enhancing Security with nftables and Beyond (4th Edition)*. 2015 Addison-Wesley Professional. ISBN 0134000021

[45] FERRARI, L. PostgreSQL 11 Server Side Programming Quick Start Guide: Effective database programming and interaction. Packt Publishing, 2019. ISBN 978-1789342222.

[46] DENT, Kyle D. *Postfix: The Definitive Guide*. O'Reilly Media, 2003. ISBN 0596002122

[47] HARIHARA Subramanian, PETHURU Raj. *Hands-On RESTful API Design Paterns and Best Practices.* Packt Publishing, 2019. ISBN 1788992664

[48] ntp.org: *Home of the Network Time Protocol.* [online]. [cit. 2021-03-29]. Dostupné z: http://www.ntp.org/

[49] BRESNAHAN, Christine, BLUM, Richard. *Linux Command Line and Shell Scripting Bible*. 2015 Wiley; ISBN 978-1118983843

[50] James A. Chambers. Raspberry Pi 4 Ubuntu 20.04 / 20.10 USB Mass Storage Boot Guide [online]. [cit. 2021-03-29]. Dostupné z: https://jamesachambers.com/raspberry-pi-4-ubuntu-20-04-usb-mass-storage-boot-guide/

[51] DOVECOT. *Dovecot manual* [online]. [cit. 2021-03-29]. Dostupné z: https://doc.dovecot.org/

[52] OVH Cloud. *Pricing.* [online]. [cit. 2021-03-29]. Dostupné z: https://www.ovhcloud.com/en/public-cloud/prices/

[53] Google Workspace. *Ceník* [online]. [cit. 2021-03-29]. Dostupné z: https://workspace.google.com/intl/cs/pricing.html

[54] Gitea. *Git with a cup of tea. A painless self-hosted Git service.* [cit. 2021-03-29]. Dostupné z: https://gitea.io/en-us/

[55] EtherPad. *Highly customizable open source online editor providing collaborative editing in really real-time* [cit. 2021-03-29]. Dostupné z: https://etherpad.org/

[56] Ondroid Wiki. *ODROID-H2/H2+ Introduction* [cit. 2021-03-29]. Dostupné z: https://wiki.odroid.com/odroid-h2/start

8 Přílohy

Vytvořená OpenAPI 3 definice použitá při testech