Stredná odborná škola elektrotechnická v Liptovskom Hrádku

J. Kollára 536, 033 01 Liptovský Hrádok

**PRAKTICKÁ ČASŤ ODBORNEJ ZLOŽKY**

**Forma: OBHAJOBA VLASTNÉHO PROJEKTU**

**Názov projektu:**

Nástroj na monitorovanie a správu serverov

Školský rok: 2024/2025

Riešiteľ: Kristína Šubjaková

Trieda: IV. B

Stredná odborná škola elektrotechnická v Liptovskom Hrádku

J. Kollára 536, 033 01 Liptovský Hrádok

**PRAKTICKÁ ČASŤ ODBORNEJ ZLOŽKY**

**Forma: OBHAJOBA VLASTNÉHO PROJEKTU**

**Názov projektu:**

Nástroj na monitorovanie a správu serverov

Školský rok: 2024/2025

Riešiteľ: Kristína Šubjaková

Trieda: IV. B

Konzultant: Ing. Martin Gabauer

**Čestné vyhlásenie**

Prehlasujem, že prácu na tému „Nástroj na monitorovanie a správu serverov“ som vypracovala samostatne, s použitím uvedenej literatúry.

V Liptovskom Hrádku, 24.4.2025

(podpis)

**Poďakovanie**

Chcem vysloviť úprimné poďakovanie všetkým, ktorí mi pri vypracovaní práce pomáhali, menovite však konzultantovi Ing. Martinovi Gabauerovi, za jeho odborné rady a pripomienky.

**Obsah**

[Úvod 5](#_Toc195566473)

[1 Súčasný stav riešenej problematiky 6](#_Toc195566474)

[1.1 Operačný systém Linux a monitorovanie serverov 6](#_Toc195566475)

[1.2 Metódy monitorovania výkonu 6](#_Toc195566476)

[1.3 Virtualizácia a Kontajnerizácia 7](#_Toc195566477)

[1.4 Python a Django vo vývoji aplikácií 8](#_Toc195566478)

[1.5 Technológie pre webový frontend 8](#_Toc195566479)

[1.6 Efektívne spracovanie a výmena dát 9](#_Toc195566480)

[1.7 Interaktívne webové rozhrania 9](#_Toc195566481)

[1.8 Dátové úložiská a správa dát 10](#_Toc195566482)

[1.9 Nástroje pre návrh, správu verzií a vývoj 10](#_Toc195566483)

[2 Ciele práce 12](#_Toc195566484)

[3 Metodika práce 13](#_Toc195566485)

[3.1 Analýza požiadaviek 13](#_Toc195566486)

[3.2 Návrh architektúry a výber technológií 14](#_Toc195566487)

[3.3 Vytvorenie projektu a databázy 15](#_Toc195566488)

[3.4 Implementácia backendu 18](#_Toc195566489)

[3.5 Návrh a implementácia používateľského rozhrania 20](#_Toc195566490)

[3.6 Nasadenie na server 21](#_Toc195566491)

[4 Výsledky práce a diskusia 24](#_Toc195566492)

[Zhrnutie 26](#_Toc195566493)

[Záver 27](#_Toc195566494)

[Zoznam použitej literatúry 28](#_Toc195566495)

# Úvod

Ako tému práce som si vybrala „Nástroj na monitorovanie a správu serverov“, pretože spoľahlivá správa serverov je nevyhnutná pre bezproblémové fungovanie IT systémov v každej modernej organizácii. Monitorovanie výkonu, stability a dostupnosti serverov pomáha včas odhaliť problémy, predchádzať výpadkom a optimalizovať využívanie systémových zdrojov. Tieto aspekty sú kľúčové pre zabezpečenie efektívneho chodu infraštruktúry a minimalizovanie rizika zlyhaní, ktoré môžu mať negatívny vplyv na prevádzku organizácií. Z tohto dôvodu som sa rozhodla vytvoriť nástroj, ktorý uľahčí monitorovanie serverov, poskytne prehľadné informácie v reálnom čase a umožní administrátorom rýchlo reagovať na potenciálne problémy. Tento nástroj pomôže zlepšiť stabilitu IT systémov a zvýšiť bezpečnosť celej infraštruktúry.

Svoj projekt začnem analýzou a definovaním kľúčových požiadaviek. Identifikujem, ktoré metriky sú dôležité na sledovanie (napríklad CPU, RAM, disková kapacita) a aké služby budem monitorovať (ako Docker a KVM). Následne vyberiem technológie, ktoré mi umožnia efektívne spracovávať a zobrazovať tieto dáta. Na backend plánujem použiť Django, zatiaľ čo na frontend JavaScript a knižnicu Chart.js na vytváranie dynamických grafov.

Po definovaní požiadaviek sa pustím do implementácie backendu. Vytvorím databázu na ukladanie údajov o výkone serverov a naprogramujem moduly na zber dát pomocou knižnice psutil. Zároveň pridám podporu pre monitorovanie a správu Docker a KVM kontajnerov, aby bolo možné ich spravovať priamo z dashboardu. Frontend bude obsahovať prehľadný dashboard s grafmi, ktoré budú zobrazovať výkonnostné údaje v reálnom čase, a nástroje na správu kontajnerov.

Po implementácii aplikácie prebehne testovanie jej funkčnosti, stability a výkonu. Overím, či všetky metriky a služby sú spoľahlivo monitorované a či systém dokáže efektívne upozorniť administrátora na problémy.

Cieľom môjho projektu je vytvoriť funkčný nástroj na monitorovanie a správu serverov, ktorý uľahčí prácu administrátorov a pomôže udržiavať stabilitu systémov. Moje očakávania sú, že projekt bude schopný efektívne monitorovať výkon serverov a spravovať dôležité služby. Do budúcna by som chcela rozšíriť tento projekt o ďalšie nástroje na sledovanie a správu serverovej infraštruktúry, aby pokryl širšie spektrum požiadaviek na správu IT prostredí.

# Súčasný stav riešenej problematiky

## Operačný systém Linux a monitorovanie serverov

„Linux patrí medzi najpoužívanejšie operačné systémy v serverovom prostredí, a to vďaka svojej stabilite, bezpečnosti a flexibilite.“[[1]](#footnote-1) Otvorený zdrojový kód umožňuje jeho používateľom vykonávať detailné úpravy a optimalizácie podľa konkrétnych požiadaviek, čo je mimoriadne dôležité pri budovaní spoľahlivých a efektívnych systémov. Zároveň ponúka množstvo výkonných nástrojov na správu a monitorovanie, ktoré pomáhajú zabezpečiť hladký chod serverov a vysoký štandard ich prevádzky. Tieto výhody, spolu s ekonomickou nenáročnosťou, robia z Linuxu ideálnu voľbu pre široké spektrum projektov – od menších serverových riešení až po veľké dátové centrá.

„V rýchlom svete informačných technológií je monitorovanie serverov viac ako kedykoľvek nevyhnutnou praxou. Zabezpečuje plynulý chod, optimalizáciu výkonu a bezpečnosť systémov. Monitorovanie zahŕňa pozorné sledovanie rôznych metrík a parametrov serverov, aplikácií a sieťových zdrojov.“[[2]](#footnote-2)

Monitorovacie metriky predstavujú kvantitatívne údaje odrážajúce stav a výkon serverov, ako je využitie CPU, pamäte, diskov a sieťových zdrojov. Sledovanie týchto metrík umožňuje identifikovať potenciálne úzke miesta a optimalizovať výkon systémov. Existujú však rôzne metódy monitorovania, ktoré sú prispôsobené špecifickým potrebám a cieľom správcov serverov.

## Metódy monitorovania výkonu

V kontexte monitorovania a správy serverov sa využívajú dve široko uznávané metodiky na zabezpečenie komplexného prehľadu o výkone systému a užívateľského zážitku: metóda USE (Využitie, Saturácia, Chyby) a metóda RED (Rýchlosť, Chyby, Dĺžka). Tieto prístupy sa zameriavajú na rôzne aspekty monitorovania a navzájom sa efektívne dopĺňajú.

„Metóda USE poskytuje prehľad o stave hardvérových zariadení. Skúma tri hlavné oblasti: využitie, ktoré meria percento času, počas ktorého je zdroj aktívny, saturáciu, ktorá zohľadňuje množstvo práce spracovávanej zdrojom, a chyby, ktoré predstavujú počet udalostí, ako sú napríklad zlyhania čítania alebo stratené paketové prenosy. Táto metodika sa najčastejšie uplatňuje pri monitorovaní procesorov, pamäte, diskov a sieťových zariadení.

Na druhej strane, metóda RED je vhodná najmä pre služby a mikroservisné prostredia. Zahŕňa rýchlosť, ktorá udáva počet žiadostí za sekundu, chyby, ktoré sledujú počet neúspešných žiadostí, a dĺžku, ktorá meria čas potrebný na spracovanie žiadostí. Tieto metriky by mali byť implementované pre všetky komponenty služby a poskytujú cenný pohľad na užívateľskú skúsenosť. RED dashboardy umožňujú definovanie dohôd o úrovni služieb (SLA) a cieľov úrovne služieb (SLO).

Pri návrhu upozornení sa odporúča zamerať sa na symptómy problémov, nie na ich príčiny. Použitie RED dashboardov na detekciu zvýšenej chybovosti alebo vysokej latencie poskytuje rýchly prehľad o možných problémoch z pohľadu užívateľov. Následne je možné analyzovať metriky USE na identifikáciu presných príčin problémov.

Kombináciou oboch prístupov je možné dosiahnuť rovnováhu medzi technickými výkonnostnými indikátormi a užívateľskou spokojnosťou.“[[3]](#footnote-3)

## Virtualizácia a Kontajnerizácia

„Virtualizácia je metodológia delenia zdrojov počítača na niekoľko výpočtových prostredí aplikovaním jedného alebo viacerých prístupov alebo technológií, hlavne rozdelenia hardvéru a softvéru, zdieľania výpočtového času, čiastočné alebo úplné simulovanie výpočtového prostredia, emulácie, kvality služieb a iných.“[[4]](#footnote-4)

Medzi populárne riešenia pre virtualizáciu patria KVM (Kernel-based Virtual Machine), VMware a Hyper-V. Virtualizácia umožňuje dynamické prideľovanie zdrojov virtuálnym strojom, čo zvyšuje flexibilitu a optimalizuje využitie dostupných hardvérových zdrojov.

„Kontajnerizácia je forma virtualizácie, kde aplikácia beží v izolovaných užívateľských priestoroch, nazývaných kontajnery, pri použití rovnakého zdieľaného operačného systému (OS).“[[5]](#footnote-5) Nástroje ako Docker umožňujú rýchle a konzistentné nasadzovanie aplikácií v rôznych prostrediach. Kontajnery zabezpečujú, že všetky komponenty aplikácie bežia v rovnakom prostredí, čo minimalizuje problémy spôsobené rozdielmi v konfiguráciách medzi rôznymi prostrediami.

## Python a Django vo vývoji aplikácií

„Python je interpretovaný, interaktívny programovací jazyk, ktorý vytvoril Guido van Rossum, pôvodne ako skriptovací jazyk pre Amoeba OS schopný systémových volaní.“[[6]](#footnote-6)

Vďaka rozsiahlej ponuke knižníc a frameworkov ako Django je Python ideálny pre vývoj backendových služieb, skriptov pre automatizáciu úloh a spracovanie dát.

„Django je vysokoúrovňový webový framework pre Python, ktorý podporuje rýchly vývoj a čistý, pragmatický dizajn.“[[7]](#footnote-7) Poskytuje množstvo nástrojov pre správu databáz, autentifikáciu používateľov a tvorbu administratívnych rozhraní. Django umožňuje implementáciu robustných autentifikačných mechanizmov, čo je nevyhnutné pre zabezpečenie prístupu k dátam a správcovským funkciám aplikácie.

## Technológie pre webový frontend

„Frontend predstavuje časť webovej aplikácie, s ktorou priamo interaguje používateľ. Zahŕňa všetky vizuálne a interaktívne prvky na stránke, ako sú tlačidlá, formuláre či navigácia.“[[8]](#footnote-8) Cieľom frontendu je zabezpečiť, aby bola aplikácia esteticky príjemná a intuitívna na používanie.

HTML (HyperText Markup Language) a CSS (Cascading Style Sheets) sú základné technológie pre tvorbu webových rozhraní. HTML definuje štruktúru a obsah webových stránok, zatiaľ čo CSS sa používa na ich vizuálnu prezentáciu a štýlovanie.

JavaScript je dynamický programovací jazyk, ktorý umožňuje vytvárať interaktívne a dynamické prvky na webových stránkach. V kombinácii s HTML a CSS tvorí základ pre vývoj moderných webových aplikácií, ktoré reagujú na akcie používateľov v reálnom čase.

Chart.js je populárna JavaScriptová knižnica na vytváranie interaktívnych grafov a vizualizácií dát na webových stránkach. Umožňuje vizualizáciu metrík, ako sú využitie CPU, pamäte, sieťových zdrojov a ďalších, čo poskytuje prehľadné a intuitívne zobrazenie systémových dát. Chart.js podporuje rôzne typy grafov, vrátane čiarových, barových, koláčových a ďalších, čo umožňuje flexibilné a prispôsobiteľné vizualizácie podľa potrieb používateľov.

## Efektívne spracovanie a výmena dát

Tento proces zahŕňa kombináciu nástrojov a technológií, ktoré umožňujú integráciu rôznych komponentov a zabezpečujú hladký tok informácií v rámci systému. Medzi najdôležitejšie technológie patria knižnica PSutil, framework Celery a dátový formát JSON

„Psutil je knižnica pre programovací jazyk Python, ktorá poskytuje rozhranie na získavanie informácií o systémových zdrojoch, ako sú využitie procesora (CPU), pamäte, diskov a sieťových pripojení“[[9]](#footnote-9). Psutil umožňuje zhromažďovanie monitorovacích metrík v reálnom čase a disponuje jednoduchým a prehľadným API, ktoré uľahčuje prístup k systémovým informáciám.

Celery je distribuovaný systém určený na spracovanie úloh, ktorý umožňuje asynchrónne vykonávanie a plánovanie ich spúšťania. Celery poskytuje možnosť delegovania náročných operácií na pozadí, čo zvyšuje responzívnosť a výkon hlavného systému, a zároveň umožňuje jednoduché rozširovanie systému o nové úlohy a funkcionality. Worker procesy v Celery sú zodpovedné za vykonávanie asynchrónnych úloh, zatiaľ čo Beat je plánovač úloh, ktorý umožňuje pravidelné spúšťanie definovaných úloh.

JSON (JavaScript Object Notation) je ľahký formát pre výmenu dát, ktorý je ľahko čitateľný pre človeka aj strojovo spracovateľný. Používa sa na komunikáciu medzi klientom a serverom, ako aj na ukladanie konfiguračných súborov, čím zabezpečuje flexibilitu a interoperabilitu systému. JSON umožňuje efektívnu serializáciu a deserializáciu dát, čo je kľúčové pre rýchlu výmenu informácií medzi rôznymi komponentmi systému.

Integrácia PSutil, Celery a JSON poskytuje robustný základ pre efektívne spracovanie a výmenu dát v monitorovacích systémoch. PSutil zabezpečuje presný zber údajov, Celery umožňuje ich rýchle spracovanie a JSON zabezpečuje ich prenos do vizualizačného rozhrania

## Interaktívne webové rozhrania

Node.js je open-source, cross-platformové runtime prostredie pre JavaScript, ktoré umožňuje spúšťanie JavaScript kódu na strane servera. Jeho asynchrónny a udalosťami riadený model je ideálny pre tvorbu škálovateľných sieťových aplikácií. „Node.js využíva udalosťami riadenú architektúru, ktorá umožňuje efektívne spracovanie asynchrónnych úloh“[[10]](#footnote-10), ako je aktualizácia metrík v reálnom čase.

Xterm.js je JavaScriptová knižnica, ktorá poskytuje terminálové emulátory priamo v prehliadači. Umožňuje integráciu interaktívnych terminálových relácií, čo zvyšuje možnosti správy serverov a poskytuje administrátorom prístup k príkazovým riadkom priamo z webového rozhrania. „Xterm.js je využívaný v mnohých svetových aplikáciách na poskytovanie kvalitných terminálových zážitkov.“[[11]](#footnote-11)

## Dátové úložiská a správa dát

Databáza je základným komponentom nástrojov na monitorovanie a správu serverov, pretože slúži na uchovávanie a správu veľkého množstva dát získaných z monitorovacích nástrojov a logov.

„PostgreSQL je výkonná relačná databáza, ktorá podporuje pokročilé funkcie ako transakcie, replikáciu a komplexné dotazy.“[[12]](#footnote-12) Slúži na ukladanie a správu dát získaných z monitorovacích nástrojov, ako aj na uchovávanie konfiguračných informácií a užívateľských dát. PostgreSQL poskytuje robustné mechanizmy zabezpečenia a integritu dát, čo je nevyhnutné pre spoľahlivé uchovávanie a spracovanie metrík a logov. Využitie PostgreSQL umožňuje efektívne dotazovanie a analýzu dát, čo je kľúčové pre poskytovanie prehľadov a reportov používateľom nástrojov.

Redis je in-memory dátové úložisko, často používané ako databáza, cache alebo broker správ. Redis slúži na rýchle ukladanie a sprístupňovanie dát v reálnom čase, čo zlepšuje výkon a efektivitu spracovania metrík. Redis umožňuje ukladanie dočasných dát, ktoré sú často prístupné, čo znižuje zaťaženie hlavnej databázy a zvyšuje rýchlosť odpovedí systému. Okrem toho Redis podporuje rôzne dátové štruktúry, ako sú reťazce, zoznamy, množiny a hashové mapy, čo umožňuje flexibilné a efektívne spravovanie rôznych typov dát.

## Nástroje pre návrh, správu verzií a vývoj

Moderný vývoj softvéru a návrh používateľských rozhraní vyžaduje efektívne nástroje, ktoré podporujú tímovú spoluprácu, správu kódu a návrh vizuálnych prvkov. Medzi kľúčové nástroje pre tieto účely patria Visual Studio Code, Git, GitHub a Figma, ktoré sú široko využívané webovými vývojármi a dizajnérmi po celom svete.

„Visual Studio Code je populárny editor zdrojového kódu od spoločnosti Microsoft.“[[13]](#footnote-13) Vyznačuje sa jednoduchým používateľským rozhraním, vysokou rýchlosťou a možnosťou rozšírenia prostredníctvom širokej ponuky doplnkov. VS Code podporuje viacero programovacích jazykov a technológií, čo z neho robí univerzálny nástroj pre vývoj aplikácií. Medzi najväčšie výhody patrí integrovaný terminál, podpora Git priamo v prostredí a množstvo doplnkov pre produktivitu, ako sú integrované ladiace nástroje alebo automatické formátovanie súborov.

„Git je systém na správu verzií, ktorý umožňuje sledovať zmeny v zdrojovom kóde, vrátiť sa k predchádzajúcim verziám a efektívne spolupracovať v tíme. Git umožňuje vývojárom paralelnú prácu na jednom projekte bez obáv zo straty dát alebo konfliktov v kóde. GitHub je cloudová platforma založená na Gite, ktorá poskytuje priestor na hosting projektov, správu repozitárov a nástroje pre tímovú spoluprácu. Pomocou funkcií, ako sú pull requesty, issue tracking a CI/CD integrácia, GitHub umožňuje efektívny vývoj a nasadzovanie aplikácií.“[[14]](#footnote-14)

Figma je moderný nástroj pre návrh používateľských rozhraní, ktorý funguje priamo v prehliadači. Umožňuje tímom pracovať na návrhoch v reálnom čase, čím eliminuje problémy spojené so synchronizáciou súborov. Figma ponúka bohaté možnosti na návrh UI/UX, prototypovanie a zdieľanie návrhov s klientmi alebo tímami.

# Ciele práce

Hlavným cieľom našej práce je vytvoriť funkčný nástroj na monitorovanie a správu serverov, ktorý poskytne administrátorom prehľadné informácie v reálnom čase, umožní efektívnu správu kľúčových služieb a pomôže udržiavať stabilitu a bezpečnosť IT infraštruktúry. Zároveň si rozšíriť vedomosti a skúsenosti v oblasti monitorovania serverov a vývoja webových aplikácií.

* Analýza požiadaviek a návrh systému na zber systémových metrík (CPU, RAM, disk, Docker, KVM) a ich ukladanie do databázy.
* Implementácia prehľadného webového rozhrania s grafmi a nástrojmi na správu kontajnerov a služieb.
* Testovanie funkčnosti, stability a výkonu aplikácie, vrátane notifikácií na potenciálne problémy a zabezpečenia možnosti budúceho rozšírenia.

Ďalším hlavným cieľom našej práce je nadobudnúť praktické zručnosti, ktoré budú užitočné v budúcom profesionálnom živote.

* Osvojiť si efektívne prezentovanie svojej práce pred odborným publikom.
* Naučiť sa písať technickú dokumentáciu v súlade so slovenskou technickou normou.

# Metodika práce

Vývoj bol realizovaný využitím viacerých moderných technológií a nástrojov. Tento proces zahŕňal návrh a implementáciu backend, tvorbu frontendového rozhrania a integráciu nástrojov na efektívne spracovanie a vizualizáciu dát.

## Analýza požiadaviek

Najskôr sme sa zamerali na analýzu existujúcich nástrojov, ako sú Prometheus, Grafana a Cockpit. Každé riešenie sme detailne preskúmali, pričom sme hodnotili ich silné a slabé stránky, ako aj možnosti ich integrácie do nášho systému.

A table with text and images

AI-generated content may be incorrect.

Obr. Prehľad funkcií dostupných nástrojov

Na základe tejto analýzy sme zostavili zoznam požiadaviek, ktoré sme potrebovali pokryť v našom riešení.

* Zber systémových metrík (CPU, RAM, disk, Docker, KVM) a ich ukladanie do databázy.
* Využitie CPU, RAM a diskových kapacít.
* Sledovanie vyťaženia hardvérových zdrojov.
* Stav behu dôležitých služieb (Nginx, Apache, PostgreSQL).
* Riadenie a monitorovanie Docker kontajnerov a KVM virtuálnych strojov.
* Ukladanie systémových logov a generovanie upozornení na základe definovaných pravidiel.

## Návrh architektúry a výber technológií

Po definovaní požiadaviek sme sa rozhodli pre dvojúrovňovú architektúru pozostávajúcu z control node a agent nodes. Control node je centrálny server, ktorý spracováva všetky údaje, analyzuje ich a vizualizuje prostredníctvom webového dashboardu. Zároveň riadi monitorovacie procesy, kontroluje zhromaždené dáta a na základe vopred definovaných pravidiel administrátora rozosiela upozornenia v prípade prekročenia nastavených limitov. Agent nodes sú skripty, ktoré bežia na jednotlivých serveroch a zbierajú metriky o výkone systému. Tieto údaje sa následne prostredníctvom šifrovaného HTTPS odosielajú v pravidelných intervaloch na control node.

Takýto model architektúry prináša viacero výhod. Škálovateľnosť systému je zabezpečená tým, že do monitorovanej siete je možné jednoducho pridať ďalšie servery jednoduchým nasadením agenta, bez nutnosti zásahu do centrálneho servera. Výpočtovo náročné operácie sú rozdelené medzi jednotlivé agent nodes, čo výrazne redukuje záťaž na hlavný server. Systém taktiež poskytuje vysokú flexibilitu, pretože umožňuje dynamické úpravy intervalov zberu údajov či rozširovanie monitorovaných metrík bez zásahu do základnej infraštruktúry. V neposlednom rade zabezpečuje zvýšenú bezpečnosť, keďže všetka komunikácia medzi agentmi a centrálnym serverom prebieha cez šifrované HTTP požiadavky s autentifikáciou pomocou API kľúčov.

Pre backend control node sme sa rozhodli použiť Django framework, ktorý poskytuje robustnú štruktúru pre vývoj webových aplikácií. Ako databázový systém sme zvolili PostgreSQL, ktorý vyniká výkonnosťou a efektívnym spracovaním veľkého množstva údajov. Celery sme využili na plánovanie asynchrónnych úloh, pričom Redis slúži ako vyrovnávacia pamäť, ktorá zrýchľuje spracovanie požiadaviek a minimalizuje oneskorenia. Pre získavanie systémových metrík sme využili knižnicu psutil, zatiaľ čo docker-py nám umožnila monitorovať a spravovať Docker kontajnery.

Na strane agent nodes sme implementovali Flask, ktorý je ľahký a umožňuje efektívnu správu HTTP požiadaviek a komunikáciu s control node. Na zabezpečenie bezpečnej komunikácie medzi agentmi a centrálnym serverom sme zaviedli šifrované API požiadavky s autentifikáciou pomocou API kľúčov.

Frontend sme vytvorili pomocou Django Templatov, ktoré umožňujú tvorbu dynamického a interaktívneho používateľského rozhrania. V kombinácii s HTML, CSS a JavaScriptom sme zabezpečili responzívnosť aplikácie a prehľadnú vizualizáciu dát pomocou knižnice Chart.js. Pred samotnou implementáciou sme používateľské rozhranie navrhli vo Figme, čo nám umožnilo efektívne testovanie a optimalizáciu UX/UI pred vývojom.

## Vytvorenie projektu a databázy

Pri vývoji aplikácie sme začali vytvorením nového Django projektu a správou verzií cez Git. Najskôr sme vytvorili virtuálne prostredie a nainštalovali Django.

A screenshot of a computer program

Description automatically generated

Obr. Okomentované príkazy na vytvorenie projektu

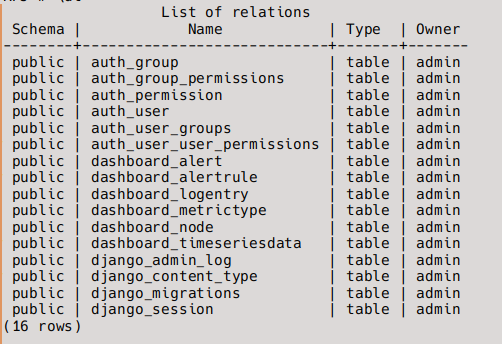
Po vytvorení projektu sme inicializovali nový Git repozitár.

A close-up of a logo

Description automatically generated

Obr. Vytvorenie repozitára git cez CLI

V rámci návrhu databázového modelu sme vytvorili niekoľko kľúčových tabuliek, ktoré zabezpečujú efektívne ukladanie a spracovanie dát potrebných pre monitorovanie systému.



Obr. Zoznam všetkých databázových tabuliek

Informácie o jednotlivých agentoch, sa zaznamenávajú v tabuľke Node. Do nej sú uložené dáta ako názov, IP adresa, typ (Control, Agent, Database), geografická poloha, prihlasovacie údaje pre SSH a aktuálny stav (online/offline). Medzi kľúčové atribúty patrí aj last\_check\_in, ktorý zaznamenáva čas poslednej komunikácie agenta s centrálnym serverom.

Ďalšou súčasťou modelu je definícia rôznych typov sledovaných metrík pomocou tabuľky MetricType. Každá metrika má jedinečný názov a jednotku merania, čo zabezpečuje konzistentné ukladanie a neskoršiu interpretáciu dát.

Časové rady nameraných hodnôt sa uchovávajú v databáze TimeSeriesData, kde každý záznam obsahuje referenciu na konkrétny uzol, typ metriky, časovú pečiatku a nameranú hodnotu. Táto štruktúra umožňuje efektívne vykonávanie analýz a vizualizácií historických dát.

Pre správu systémových logov sme vytvorili tabuľku LogEntry, ktorá uchováva informácie o jednotlivých záznamoch. Každý log obsahuje časovú pečiatku, názov hostiteľa, názov služby, prioritu logu a samotnú správu. Záznamy sú usporiadané podľa priority, čo umožňuje rýchle vyhľadávanie kritických udalostí.

Pre automatizáciu vytvárania upozornení sme navrhli tabuľku AlertRule, ktorá umožňuje definovať pravidlá na sledovanie metrík. Používateľ môže nastaviť prahovú hodnotu, spôsob porovnania (> , <, >=, <=, =, !=), závažnosť upozornenia a dobu cooldown, ktorá zabraňuje opakovanému spúšťaniu rovnakého upozornenia v krátkom čase. Pri prekročení nastavených limitov sa automaticky generuje upozornenie, ktoré sa ukladá do tabuľky Alert a prípadne odosiela notifikácie administrátorom. Každé upozornenie je spojené s konkrétnym uzlom a typom metriky, obsahuje čas vzniku, úroveň závažnosti (info, warning, critical) a samotnú správu.

Pri optimalizácii databáz orientovaných na časové rady sa čoraz častejšie využíva technológia Timescale. Ide o rozšírenie PostgreSQL, ktoré umožňuje vytvárať tzv. hypertabulky—špeciálne optimalizované tabuľky pre veľké objemy časových dát. Timescale efektívne rozdeľuje údaje do menších segmentov podľa časového intervalu, čím sa znižuje záťaž na dotazy a zvyšuje rýchlosť agregácií. Navyše, ponúka funkcie ako kontinuálne agregáty a kompresiu dát, ktoré pomáhajú optimalizovať využitie úložiska a umožňujú prakticky bezproblémové škálovanie dátových modelov, čo je kľúčové pri monitorovaní a analýze časovo závislých meraní.

A graph of different colored bars

AI-generated content may be incorrect.

Obr. Porovnanie výkonnosti databázových dotazov pred a po optimalizácii

Tento databázový model poskytuje stabilnú a flexibilnú základňu pre efektívne monitorovanie systémov, uchovávanie historických dát a spracovanie upozornení v reálnom čase.

## Implementácia backendu

Backend je rozdelený na viacero častí, pričom každá zabezpečuje konkrétnu funkcionalitu systému.

Zber dát z agent nodes prebieha pomocou Django management komandu, ktorý je automatizovaný cez Celery task. Control node pravidelne spúšťa tento príkaz, aby získal aktuálne metriky zo všetkých agent nodes. Po prijatí údajov sa tieto dáta validujú a ukladajú do databázy.

A screenshot of a computer program

Description automatically generated

Obr. Zjednodušený kód Django komandu

Celery je použitý na pravidelné spúšťanie management komandu, čím sa zabezpečí, že dáta budú zbierané automaticky bez manuálneho zásahu

A white background with green text

Description automatically generated

Obr. Zjednodušený kód automatizovaného Celery tasku

Keď je nový údaj uložený do databázy, Django signals zabezpečí, že sa automaticky skontrolujú všetky relevantné pravidlá pre alerty a v prípade potreby sa vygeneruje upozornenie.

A screenshot of a computer code

Description automatically generated

Obr. Zjednodušený kód na kontrolu pravidiel

Tento mechanizmus zabezpečuje, že alerty sú spracovávané okamžite po prijatí nových dát, bez potreby manuálnych kontrol alebo plánovaných úloh, čím sa dosahuje efektívna a real-time detekcia problémov v systéme. Po spracovaní alertu sa odošle upozornenie administrátorovi prostredníctvom e-mailu a súčasne sa informácia o incidente odosiela na Discord server pomocou webhooku, čím sa zabezpečí rýchle informovanie zodpovedných osôb.

A screen shot of a computer code

Description automatically generated

Obr. Vzorový kód pre odosielanie upozornení

## Návrh a implementácia používateľského rozhrania

Používateľské rozhranie aplikácie bolo navrhnuté s dôrazom na prehľadnosť a intuitívne ovládanie. Hlavným cieľom bolo poskytnúť administrátorom serverov rýchly prístup k dôležitým informáciám a nástrojom na správu infraštruktúry. Dizajn vychádza z moderných princípov UX/UI a bol vytvorený pomocou nástroja Figma.

Používateľské rozhranie je štruktúrovane rozdelené do niekoľko hlavných sekcií.

* Dashboard - Hlavná stránka obsahuje prehľad výkonnostných metrík serverov, aktuálny stav sledovaných služieb (Docker, KVM, webové servery) a vizualizáciu historických údajov.
* Docker Monitor - Sekcia určená na monitorovanie a správu Docker kontajnerov s možnosťou ich spúšťania, zastavenia alebo reštartovania.
* Webový terminál - Rozhranie umožňujúce interakciu so serverom prostredníctvom emulovaného terminálu priamo v prehliadači.
* Systémové logy - Stránka zobrazujúca logy serverov s možnosťou filtrovania podľa dátumu, úrovne dôležitosti a obsahu.
* Správca alertov - Ponúka rozhranie na správu upozornení s možnosťou detailnejšej analýzy za zvolené obdobie.
* Nodes - Sekcia zobrazujúca stav pripojených nodov spolu s informáciami o control node. Zároveň umožňuje párovanie nových nodov a ich odstraňovanie.
* Nastavenia - Brána do Django admin rozhrania.

Na generovanie dynamických HTML stránok využíva aplikácia Django Templates, ktoré umožňujú vkladanie dynamického obsahu do HTML. Django templaty poskytujú nástroje na správu obsahu, cykly a podmienky, čo uľahčuje prispôsobenie rozhrania podľa aktuálnych údajov.



Obr. Príklad použitia Django templatov

Pre každú sekciu je vytvorený vlastný Django template a taktiež aj pre navigačné menu, ktoré užívateľovi umožňuje prepínanie medzi sekciami.

A screenshot of a computer

Description automatically generated

Obr. Priečinok s templatami

Aplikácia využíva na vizualizáciu údajov knižnicu Chart.js, ktorá umožňuje vytvárať interaktívne grafy zobrazujúce rôzne metriky serverov, ako napríklad využitie CPU, pamäte alebo sieťovej prevádzky. Chart.js poskytuje širokú škálu prispôsobiteľných grafov, ako sú čiarové, stĺpcové alebo koláčové grafy, čo umožňuje efektívnu analýzu údajov.

A screenshot of a computer code

Description automatically generated

Obr. Príklad použitia Chart.js v template

Vďaka týmto technológiám aplikácia zabezpečuje dynamické zobrazovanie údajov a efektívne spracovanie informácií pre administrátorov serverov.

## Nasadenie na server

Pre správne fungovanie aplikácie v produkčnom prostredí je potrebné zabezpečiť jej stabilné nasadenie na server.

Pred samotným nasadením je potrebné nakonfigurovať server, ktorý bude hostovať aplikáciu. Použitý bol server s operačným systémom Ubuntu Server a nasledovné komponenty.

* NGINX ako reverzný proxy server.
* Gunicorn na spustenie Django aplikácie.
* PostgreSQL ako databázový systém.
* Redis na ukladanie cache a správu frontov úloh.
* Celery na spracovanie asynchrónnych úloh.

Proces nasadzovania sa skladá z nasledujúcich krokov. Najprv je na serveri potrebné vytvoriť virtuálne prostredie pre Python a nainštalovať všetky požadované balíčky.

A white background with black text

Description automatically generated

Obr. Použité príkazy na vytvorenie prostredia

Ďalším krokom je konfigurácia databázy PostgreSQL čo môžeme uskutočniť vytvorením nového používateľa a databázy.

A screen shot of a computer code

Description automatically generated

Obr. SQL príkazy pre vytvorenie používateľa a databázy

Teraz vytvorím tabuľky v databáze pomocou Django management príkazov a tiež pozbieram všetky statické súbory aplikácie.

A white background with black and red lines

Description automatically generated with medium confidence

Obr. Django management príkazy

Aplikácia sa spúšťa cez Gunicorn, ktorý zabezpečuje jej beh.



Obr. Príkaz pre spustenie Gunicornu

Následne musíme konfigurovať NGINX ako reverzný proxy server, ktorý presmerováva požiadavky na Gunicorn. Konfigurácia v /etc/nginx/sites-available/monitoring vyzerá nasledovne.

A close-up of a logo

Description automatically generated

Obr. Vzorová konfigurácia NGINX

Po úprave konfigurácie je ju potrebné aktivovať a reštartovať NGINX.



Obr. Príkazy pre vytvorenie symbolického link a reštartovania NGINXu

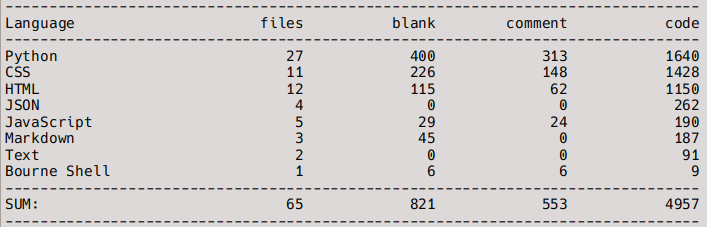
Nakoniec ostáva už iba spustiť automatizované Celery procesy pomocou skriptu run.sh.



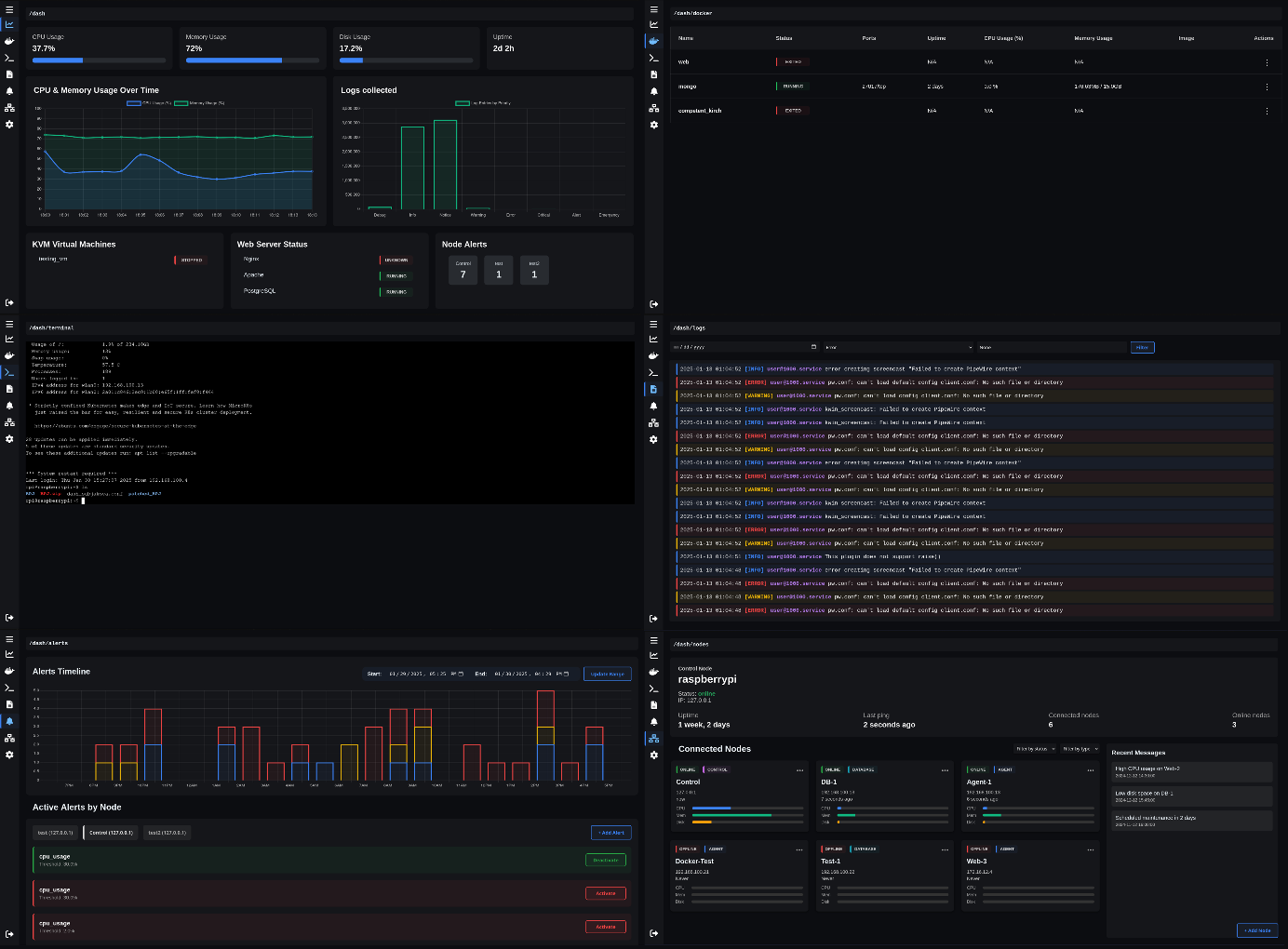
Obr. Príkazy pre spustenie Celery procesov

# Výsledky práce a diskusia

Výsledkom našej práce je funkčný a prehľadný nástroj na monitorovanie a správu serverov. Nástroj sa nám taktiež úspešne nasadiť a je plne funkčný na našej webovej doméne. Webové rozhranie nástroja sa skladá z šiestich hlavných pod stránok. Zdrojový kód nástroja je uložený do 65 súborov ktoré dokopy obsahujú viac ako 4900 riadkov kódu. Počas práce sme vykonali vyše 200 commitov v našom github repozitári.



Obr. Tabuľka projektových štatistík



Obr. Koláž vytvorených hlavných podstránok

V našej práci sme vytvorili systém na monitorovanie a správu serverov s prihliadnutím na riešenia, ako sú Prometheus či Copilot. V porovnaní s týmito nástrojmi sme kládli dôraz na jednoduchšie používateľské rozhranie, ľahšiu správu kontajnerov a vyššiu možnosť prispôsobenia. Naše výsledky preukázali, že systém je vhodný pre menšie a stredné infraštruktúry, pričom znižuje čas riešenia incidentov a zvyšuje efektivitu dohľadu.

Pri porovnaní s teoretickými poznatkami sme dospeli k záveru, že moderné monitorovacie platformy musia byť škálovateľné a modulárne. Naša aplikácia to spĺňa vďaka využitiu Django, Celery a Redis na spracovanie dát v reálnom čase a knižnice Chart.js na vizualizáciu. Z hľadiska cieľov definovaných v úvode sme sa zamerali na sledovanie systémových metrík (CPU, RAM, disk) a správu Docker kontajnerov, čím sme vyhoveli hlavným požiadavkám na flexibilitu a prehľad.

V budúcnosti by bolo prínosné zaviesť rozšírenú analýzu historických dát a podporu ďalších virtualizačných platforiem. Takéto vylepšenia by ďalej zvýšili hodnotu systému a priblížili ho k najlepším monitorovacím riešeniam na trhu, pričom by zachovali jeho používateľskú prívetivosť a možnosť prispôsobenia.

# Zhrnutie

Tento projekt sa zaoberá návrhom a implementáciou systému na monitorovanie a správu serverov. V úvodných častiach je zdôvodnená potreba takéhoto nástroja a vykonaná analýza existujúcich riešení. V texte sú predstavené metódy monitorovania výkonu serverov a opísané základné princípy fungovania Linuxu, virtualizácie a kontajnerizácie.

Jadrom práce je vývoj aplikácie v jazyku Python za použitia Django frameworku, pričom dôraz sa kladie na efektívnu správu dát, interaktívne webové rozhranie a vizualizáciu metrík prostredníctvom knižnice Chart.js. V metodickej časti sú detailne popísané procesy analýzy požiadaviek, návrhu architektúry, výberu technológií a samotnej implementácie backendu aj frontendu. Súčasťou riešenia je aj postup nasadenia aplikácie na produkčný server.

Výsledkom projektu je plne funkčný systém, ktorý umožňuje monitorovanie kľúčových parametrov (CPU, RAM, disková kapacita a Docker kontajnery), správu logov a interakciu so systémom cez webový terminál. Projekt ponúka podrobný popis implementácie a nasadenia aplikácie, čo vytvára priestor pre budúce rozšírenie o pokročilé analytické nástroje a rozmanité monitorovacie funkcie.

# Záver

V rámci tejto práce sa podarilo vytvoriť nástroj na monitorovanie a správu serverov, ktorý poskytuje administrátorom prehľad o infraštruktúre v reálnom čase. Splnili sme stanovené ciele – aplikácia umožňuje sledovať dôležité systémové metriky (CPU, RAM, disk, Docker, KVM), spravovať kontajnery, uchovávať systémové logy a upozorňovať na prípadné problémy. Využitím moderných technológií (Django, Celery, Redis, Chart.js) vznikol škálovateľný a rozšíriteľný systém, ktorý dokáže pružne reagovať na rôznorodé prevádzkové scenáre.

Hlavným prínosom projektu je zefektívnenie správy serverov, skrátenie času na reakciu pri incidentoch a zvýšenie prehľadnosti pri dohľade nad infraštruktúrou. V praxi môže tento nástroj poslúžiť nielen profesionálnym administrátorom, ale aj jednotlivcom spravujúcim vlastné servery, ktorí potrebujú spoľahlivo monitorovať stav systémových zdrojov a rýchlo zasahovať pri vzniku problémov.

Priestor na ďalší vývoj sa otvára najmä v rozšírení podpory pre ďalšie virtualizačné a kontajnerizačné platformy, pokročilú analýzu historických dát či integráciu s externými nástrojmi na správu incidentov. Do budúcnosti je dôležité zamerať sa na zvýšenie bezpečnosti a optimalizáciu výkonu pri spracovaní veľkých objemov údajov. Získané poznatky a skúsenosti v oblasti infraštruktúrneho monitorovania a vývoja webových aplikácií vytvárajú pevný základ pre ďalšie zdokonaľovanie tohto riešenia.

# Zoznam použitej literatúry

1. **CODERAMA:** IT slovník – Frontend. [cit. 2024-15-12]. Dostupné na internete: https://coderama.com/slovnik/frontend
2. **DJANGO:** Meet Django. [cit. 2024-13-12]. Dostupné na internete: https://www.djangoproject.com/
3. **Dominique M.:** Monitorovanie RDP. [cit. 2024-13-12]. Dostupné na internete: https://tsplus.net/sk/server-monitoring/blog/rdp-monitoring-enhance-server-performance-security/
4. **EDU TRAININGS:** Čo je kontajnerizácia? [cit. 2024-12-12]. Dostupné na internete: https://edutrainings.sk/area/kontajnerizacia/
5. **GITHUB:** About GitHub and Git. [cit. 2024-18-12]. Dostupné na internete: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git
6. **GRAFANA LABS:** Best practices for building effective dashboards. [cit. 2024-13-12]. Dostupné na internete: https://grafana.com/docs/grafana/latest/dashboards/build-dashboards/best-practices/
7. **Hatina M.:** Čo je virtualizácia? [cit. 2024-12-12]. Dostupné na internete: https://www.fi.muni.cz/~kas/pv090/referaty/2013-jaro/virt.html
8. **Lane R.:** Event-Driven Programming in Node.js. [cit. 2024-17-12]. Dostupné na internete: https://www.digitalocean.com/community/tutorials/nodejs-event-driven-programming
9. **MICROSOFT:** Visual Studio Code. [cit. 2024-16-12]. Dostupné na internete: https://learn.microsoft.com/en-us/shows/visual-studio-code/
10. **POSTGRESQL:** What is PostgreSQL? [cit. 2024-16-12]. Dostupné na internete: https://www.postgresql.org/about/
11. **PSUTIL:** Psutil documentation. [cit. 2024-17-12]. Dostupné na internete: https://psutil.readthedocs.io/en/latest/
12. **WEBSUPPORT:** Čo je to Linux server. [cit. 2024-11-12]. Dostupné na internete: https://www.websupport.sk/podpora/kb/co-je-to-linux-server/
13. **WIKIPEDIA:** Python (programovací jazyk). [cit. 2024-12-12]. Dostupné na internete: https://sk.wikipedia.org/wiki/Python\_(programovac%C3%AD\_jazyk)
14. **XTERM:** Real-world uses. [cit. 2024-17-12]. Dostupné na internete: https://xtermjs.org/

1. WEBSUPPORT: *Čo je to Linux server*. [cit. 2024-11-12]. Dostupné na internete:

   <https://www.websupport.sk/podpora/kb/co-je-to-linux-server/> [↑](#footnote-ref-1)
2. Dominique M.: *Monitorovanie RDP.* [cit. 2024-13-12]. Dostupné na internete:

   <https://tsplus.net/sk/server-monitoring/blog/rdp-monitoring-enhance-server-performance-security/> [↑](#footnote-ref-2)
3. GRAFANA LABS: *Best practices for building effective dashboards*. [cit. 2024-13-12]. Dostupné na

   internete: <https://grafana.com/docs/grafana/latest/dashboards/build-dashboards/best-practices/> [↑](#footnote-ref-3)
4. Hatina M.: *Čo je virtualizácia*? [cit. 2024-12-12]. Dostupné na internete:

   <https://www.fi.muni.cz/~kas/pv090/referaty/2013-jaro/virt.html> [↑](#footnote-ref-4)
5. EDU TRAININGS: *Čo je kontajnerizácia?* [cit. 2024-12-12]. Dostupné na internete:

   <https://edutrainings.sk/area/kontajnerizacia/> [↑](#footnote-ref-5)
6. WIKIPEDIA: *Python (programovací jazyk).* [cit. 2024-12-12]. Dostupné na internete:

   <https://sk.wikipedia.org/wiki/Python\_(programovac%C3%AD\_jazyk)> [↑](#footnote-ref-6)
7. DJANGO: Meet Django. [cit. 2024-13-12]. Dostupné na internete: <https://www.djangoproject.com/> [↑](#footnote-ref-7)
8. CODERAMA*: IT slovník – Frontend*. [cit. 2024-15-12]. Dostupné na internete:

   <https://coderama.com/slovnik/frontend> [↑](#footnote-ref-8)
9. PSUTIL: *Psutil documentation.* [cit. 2024-17-12]. Dostupné na internete:

   <https://psutil.readthedocs.io/en/latest/> [↑](#footnote-ref-9)
10. Lane R.: *Event-Driven Programming in Node.js*. [cit. 2024-17-12]. Dostupné na internete:

    <https://www.digitalocean.com/community/tutorials/nodejs-event-driven-programming> [↑](#footnote-ref-10)
11. XTERM: *Real-world uses*. [cit. 2024-17-12]. Dostupné na internete: <https://xtermjs.org/> [↑](#footnote-ref-11)
12. POSTGRESQL: *What is PostgreSQL?* [cit. 2024-16-12]. Dostupné na internete:

    <https://www.postgresql.org/about/> [↑](#footnote-ref-12)
13. MICROSOFT: *Visual Studio Code*. [cit. 2024-16-12]. Dostupné na internete:

    <https://learn.microsoft.com/en-us/shows/visual-studio-code/> [↑](#footnote-ref-13)
14. GITHUB: *About GitHub and Git*. [cit. 2024-18-12]. Dostupné na internete:

    <https://docs.github.com/en/get-started/start-your-journey/about-github-and-git> [↑](#footnote-ref-14)