

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект програмного агента для контролю до-
ступу до ресурсів та виявлення аномальної активності у
локальній мережі»

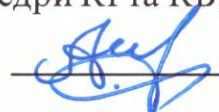
Виконав: студент 4-го курсу групи КБк-22-1
спеціальності 125 «Кібербезпека»

 Владислав ЗМІЄВСЬКИЙ

Керівник: завідувач кафедри КІ та КБ, к.т.н., доцент,

 Андрій ЄФІМЕНКО

Рецензент: доцент кафедри КІ та КБ, к.т.н.,

 Олексій ШЕЛУХА

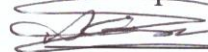
Житомир – 2025

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко
3 березня 2025 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Змієвський Владислав Віталійович

Тема роботи: «Проект програмного агента для контролю доступу до ресурсів та виявлення аномальної активності у локальній мережі»

затверджена Наказом університету від 3 березня 2025 р. № 80/с

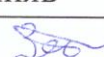
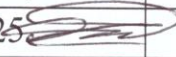
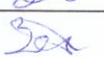
Термін здачі студентом закінченої роботи 27 червня 2025 р.

Вихідні дані роботи: предмет дослідження — моніторинг мережевих ресурсів і виявлення аномалій, об'єкт — програмний агент моніторингу, реалізований з використанням Python, HTTPS і Telegram.

Обладнання та програмне забезпечення:

1. Емулятор GNS3
2. ОС Linux Ubuntu, Windows Server
3. ПО PyCharm Community

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Єфіменко А.А.	10.05.2025 	10.05.2025 
2	Єфіменко А.А.	25.05.2025 	25.05.2025 
3	Єфіменко А.А.	03.06.2025 	03.06.2025 

Студент  Владислав ЗМІЄВСЬКИЙ

Керівник  Андрій ЄФІМЕНКО

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	15.05.2025 – 20.05.2025	Виконано
2	Огляд існуючих підходів, методів, технологій, аналогів	21.05.2025 – 26.05.2025	Виконано
3	Вибір мови програмування	26.05.2025 – 27.05.2025	Виконано
4	Створення плану офісу	27.05.2025 – 28.05.2025	Виконано
5	Створення LAN в GNS	27.05.2025 – 28.05.2025	Виконано
6	Розробка схеми адресації пристроїв мережі	28.05.2025 – 29.05.2025	Виконано
7	Налаштування мережі: маршрутизатору та комутаторів	29.05.2025 – 31.05.2025	Виконано
8	Налаштування Flask, Nginx, HTTPS/сертифікату	01.06.2025 – 07.06.2025	Виконано
9	Інтеграція з Telegram-ботом та базою даних SQLite	08.06.2025 – 13.06.2025	Виконано
10	Імітація критичних навантажень і перевірка надсилання сповіщень	14.06.2025 – 18.06.2025	Виконано
11	Інтеграція з Grafana	14.06.2025 – 18.06.2025	Виконано

Студент  Владислав ЗМІЄВСЬКИЙ

Керівник  Андрій ЄФІМЕНКО

АНОТАЦІЯ

У межах кваліфікаційної роботи розроблено програмний агент моніторингу мережевих ресурсів з можливістю виявлення аномальної активності та надсилання сповіщень через Telegram-бота. Агент реалізовано мовою програмування Python із використанням Flask, SQLite, REST API та NGINX з підтримкою HTTPS. Візуалізація метрик реалізована через інтеграцію з Grafana.

Обсяг пояснювальної записки складає 43 сторінки, вона містить 43 рисунки, 4 таблиці, список використаних джерел нараховує 21 найменування та розміщений на 2 сторінках.

Метою кваліфікаційної роботи є розробка функціонального програмного засобу для збору, обробки та візуалізації мережевих метрик, а також для виявлення потенційно шкідливої або нетипової активності у локальній мережі підприємства.

У результаті виконання роботи визначено основні вимоги до агента моніторингу, проаналізовано сучасні підходи до збору та візуалізації метрик, розглянуто особливості реалізації безпечного обміну даними через HTTPS, показано розробку REST API для взаємодії клієнта та сервера, наведено результати тестування системи в умовах імітованих навантажень.

Ключові слова: ПРОГРАМНИЙ АГЕНТ, МОНІТОРИНГ, МЕРЕЖА, PYTHON, HTTPS, TELEGRAM, GRAFANA, АНОМАЛІЇ, NAT, REST API, FLASK, SQLITE, NGINX, БЕЗПЕКА, СИСТЕМА СПОВІЩЕНЬ.

					ККІКБ.КР.Б – 125 – 4 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Проект програмного агента для контролю доступу до ресурсів та виявлення аномальної активності у локальній мережі	Літ.	Арк.	Аркушів
Розроб.		Змієвський В.В.		26.06.25				
Керівник		Єфіменко А.А.		26.06.25			4	54
Реценз.		Шелуха О.О.		26.06.25		ФІКТ, Гр. КБк-22-1		
Н. контр.		Єфіменко А.А.		26.06.25				
Затверд.		Єфіменко А.А.		26.06.25				

ANNOTATION

As part of the bachelor's qualification project, a software agent was developed for monitoring network resources, detecting anomalous activity, and sending notifications via a Telegram bot. The agent is implemented in the Python programming language using Flask, SQLite, REST API, and NGINX with HTTPS support. Metric visualization is provided through integration with Grafana.

The explanatory report consists of 43 pages and includes 43 figures, 4 tables, and a list of 21 references spread over 2 pages.

The aim of the qualification project is to develop a functional software tool for collecting, processing, and visualizing network metrics, as well as detecting potentially malicious or abnormal activity within an enterprise's local network.

As a result of the work, the main requirements for the monitoring agent were defined; modern approaches to metric collection and visualization were analyzed; the specifics of secure data exchange via HTTPS were examined; the development of a REST API for client-server interaction was demonstrated; and the results of system testing under simulated load conditions were presented.

Keywords: SOFTWARE AGENT, MONITORING, NETWORK, PYTHON, HTTPS, TELEGRAM, GRAFANA, ANOMALIES, NAT, REST API, FLASK, SQLITE, NGINX, CERTBOT, SECURITY, NOTIFICATION SYSTEM.

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП	8
РОЗДІЛ 1 АНАЛІЗ АНАЛОГІВ ТА РОЗРОБКА СТРУКТУРНОЇ СХЕМИ.	10
1.1 Zabbix.....	10
1.2 Prometheus	10
1.3 Порівняльний аналіз власного рішення з існуючими системами моніторингу.....	12
РОЗДІЛ 2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ МОНІТОРИНГУ МЕРЕЖІ ТА КОНТРОЛЮ ДОСТУПУ ДО РЕСУРСІВ.....	14
2.1 Створення плану офісу	14
2.2 Побудова моделі LAN у середовищі GNS3	15
2.3 Вибір мови програмування для розробки програмного агента.....	20
2.4 Вибір мережевих протоколів для обміну даними.....	22
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСОБУ	25
3.1 Архітектура програмного агента	25
3.2 Опис REST API-запитів до сервера моніторингу	29
3.3 Виявлення аномальної активності.....	32
3.4 Візуалізація та представлення даних у Grafana	35
ВИСНОВКИ.....	40
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	41
ДОДАТКИ.....	43

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

LAN – Local Area Network, локальна обчислювальна мережа

NAT – Network Address Translation

PAT – Port Address Translation

DHCP – Dynamic Host Configuration Protocol

HTTP – HyperText Transfer Protocol

TCP – Transmission Control Protocol

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У сучасних умовах жоден офіс вже не може обійтися без хоча б одного комп'ютера — багатофункціонального та зручного пристрою для роботи з інформацією, обробки даних та виконання різноманітних завдань. Однак зростання кількості комп'ютерних систем та локальних мереж тягне за собою збільшення потенційних загроз для інформаційної безпеки, зокрема — несанкціонованого доступу до ресурсів та появи аномальної активності в мережі. Подібні інциденти можуть суттєво вплинути на стабільність роботи програмного та апаратного забезпечення, призвести до втрати або пошкодження важливих даних, а також знизити продуктивність праці користувачів.

Актуальність теми. Ресурси комп'ютерних систем відіграють ключову роль у виконанні щоденних завдань користувачів. Їх нестача або неправильне використання безпосередньо впливає на ефективність роботи як окремих співробітників, так і підприємства в цілому. Відстеження змін стану апаратних та програмних ресурсів дозволяє своєчасно реагувати на загрози, оптимізувати конфігурацію обладнання та програмних засобів, а також приймати обґрунтовані рішення щодо розвитку інфраструктури. Особливої актуальності набуває створення програмних агентів, здатних автоматично контролювати доступ до ресурсів локальної мережі та виявляти аномальні дії, які можуть свідчити про спроби вторгнення або внутрішні проблеми системи.

Метою кваліфікаційної роботи є розробка програмного агента моніторингу для контролю доступу до ресурсів та виявлення аномальної активності у локальній мережі.

Визначена мета обумовлює наступні завдання:

- створення плану офісу;
- побудова моделі мережі в середовищі GNS3;
- розробка модуля збору даних;
- розробка модуля обробки зібраної інформації;
- розробка модуля формування та надсилання екстрених оповіщень;

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

- створення Telegram-бота для сповіщення користувачів.
- інтеграція Grafana для візуалізації даних

Об’єктом дослідження є процес забезпечення інформаційної безпеки локальної комп’ютерної мережі організації.

Предметом дослідження є програмні засоби та методи моніторингу мережевого трафіку, виявлення аномальної активності, а також технології реалізації програмних агентів для контролю доступу до інформаційних ресурсів.

У процесі виконання кваліфікаційної роботи було розроблено проект програмного агента, що забезпечує моніторинг мережевих ресурсів та виявлення аномальних дій у локальній мережі з можливістю надсилання сповіщень користувачам через Telegram-бота. Крім того, до проєкту було додано інтеграцію з Grafana, що дозволяє відображати отримані метрики у зручному вигляді за допомогою дашбордів та графіків.

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 АНАЛІЗ АНАЛОГІВ ТА РОЗРОБКА СТРУКТУРНОЇ СХЕМИ

У цьому розділі розглянемо існуючі альтернативи використання агентів моніторингу у локальній мережі. Ми розглянемо такі популярні рішення як Zabbix та Prometheus.

1.1 Zabbix

Доволі поширеною та однією з найкращих систем моніторингу є Zabbix. Дана програма має повністю відкритий код, тож є безкоштовною. Програма є універсальною в плані ресурсів моніторингу: сервери, мережі, програми, бази даних та інші пристрої. Вона має широкий спектр функцій для відстеження метрик продуктивності, моніторингу доступності служб, аналізу відмов, автоматизації оповіщень та багато іншого [1].

За допомогою Zabbix адміністратори можуть налаштовувати моніторинг різноманітних параметрів, таких як стан системи, навантаження сервера, доступність мережі, використання ресурсів тощо. Вона також надає розширені можливості для налаштування оповіщень, що дозволяє оперативно реагувати на відмови або проблеми у системі [2].

Переваги:

- здійснює моніторинг як програм так і будь-яких пристроїв ;
- підтримує різні версії протоколів моніторингу;
- має велику кількість шаблонів оформлення і метрик для збору;
- є можливість налаштування тригерів для різних умов;

Недоліки:

- через велику кількість доступних функцій може бути складним для нових користувачів;

1.2 Prometheus

Prometheus — це інструмент з відкритим кодом для моніторингу систем і сповіщень, який спочатку був створений у компанії SoundCloud. З моменту

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

свого створення в 2012 році багато компаній та організацій почали використовувати Prometheus, і зараз цей проєкт має дуже активну спільноту розробників і користувачів [3].

На сьогодні Prometheus є незалежним проєктом з відкритим кодом і підтримується автономно, без прив'язки до жодної компанії.

Prometheus збирає та зберігає свої метрики у вигляді часових рядів (time series data), тобто кожен запис містить інформацію про метрику разом із часо-вою міткою, коли її було зафіксовано, а також додатковими ключзначеннями, які називаються мітками (labels).

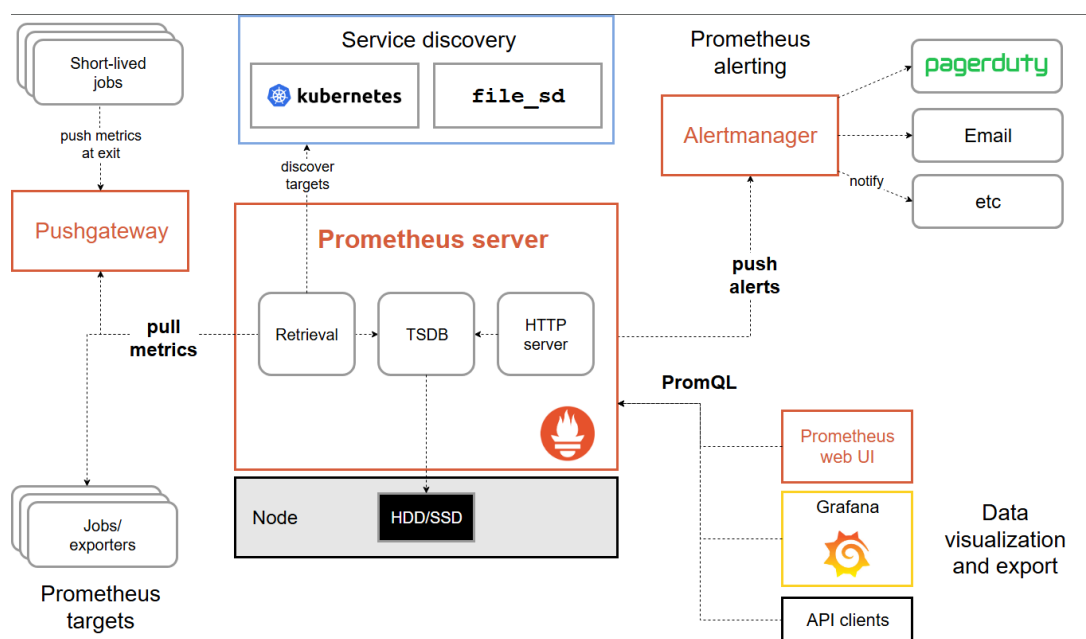


Рисунок 1.1 – Архітектура Prometheus [3]

Prometheus збирає (scrapes) метрики з інструментованих сервісів або через Push Gateway для короткотривалих завдань. Усі отримані вибірки зберігаються локально. На основі цих даних Prometheus виконує правила для агрегації, створення нових часових рядів або генерації сповіщень. Для візуалізації зібраних даних можна використовувати Grafana або інші APIклієнти.

Переваги:

- сервер може бути встановленим на різні операційні системи ;
- підтримує візуалізацію даних в Grafana, в якій доступні різні графіки і дашборди;

- є можливість налаштування екстрених сповіщень;

Недоліки:

- налаштування серверу відбувається через редагування конфігураційних файлів;

1.3 Порівняльний аналіз власного рішення з існуючими системами моніторингу

У даному підрозділі проведено порівняння розробленої системи моніторингу із відомими аналогами — такими як Zabbix та Prometheus. Метою цього аналізу є визначення переваг і обмежень власної програми в контексті поставлених завдань, зокрема: організації збору системних метрик, виявлення аномальної активності, оповіщення адміністратора та візуалізації даних у Grafana. Основною вимогою до розробленої системи було створення легкого, автономного та кастомізованого рішення без потреби використання складних або надлишкових за функціональністю систем.

Таблиця 1.1

Порівняльна характеристика аналогів

Характеристика	Власне ПО для контролю доступу	Zabbix	Prometheus + Grafana
Основна мета	Локальний моніторинг з Telegram alert	Розширений моніторинг систем, мереж, сервісів	Time Series моніторинг ресурсів, метрик
Сховище даних	SQLite	MySQL / PostgreSQL	Власна TSDB (Time Series DB)
Алерти	Telegram через власну логіку	E-mail, Telegram, SMS, script	Alertmanager: Telegram, Email, Slack
Візуалізація	Grafana (гнучка, красиві дашборди)	Вбудовані веб-звіти та графіки	Grafana (гнучка, красиві дашборди)
Агенти на хості	Python-агент (кастомний)	Zabbix Agent / Agent2	Node Exporter, custom exporter
Підтримка метрик	CPU, RAM, диск (розширюється вручну)	Дуже багато (диски, служби, процеси, SNMP, IPMI)	Всі числові time-series метрики

Висновки до першого розділу

Під час формулювання вимог до власного рішення основними завданнями було визначено: мінімізацію ресурсного навантаження на клієнтські хости, простоту розгортання, забезпечення контролю за основними показниками навантаження системи (CPU, RAM, Disk), можливість виявлення перевантажень у реальному часі та надсилання сповіщень адміністратору. Окрім того, передбачалася інтеграція з Grafana для візуалізації метрик без використання складних зовнішніх компонентів збору даних, таких як Node Exporter або Prometheus.

Опис виявлених переваг та недоліків основних використаних технологій у межах власного проєкту:

Переваги:

- для розгортання достатньо Python-скрипта, SQLite та Flask, що підходить для малих або локальних мереж
- агент споживає мало ресурсів на клієнтській машині, на відміну від повноцінних агентів Zabbix або експортерів для Prometheus.
- код Python можна швидко змінити або доповнити новими метриками чи алгоритмами обробки.
- SQLite дозволяє зберігати дані локально без розгортання додаткових серверів БД;

Недоліки:

- власне рішення погано підходить для моніторингу десятків або сотень хостів, де Zabbix чи Prometheus вже мають централізовані сервери збору, агрегації, шардінгу.
- SQLite не підходить для великих обсягів метрик або довготривалого зберігання;
- Zabbix та Prometheus мають широку екосистему (плагіни, exporters, шаблони), які у власному рішенні потрібно створювати самостійно.

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				14
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 2.3 – 3d-план приміщення офісу

На рисунках 2.1.-2.3 представлена схема офісного простору з поділом на кімнати та робочі зони [4]. Ось опис кабінетів і зон, які можна виділити:

- зона відпочинку з диванами, столами й телевізором;
- кімнати з робочими зонами для працівників, Open Space;
- конференц-зал з великим столом, проектором, телевізором;
- кімната адміністратора з ядром мережі (серверним або технічним обладнанням);
- кімната для співбесід, переговорна з круглим столом і кількома кріслами;
- коридор, зона очікування (приймальня);
- санвузол / технічне приміщення.

2.2 Побудова моделі LAN у середовищі GNS3

Створена мережа повинна включати основні пристрої, які застосовані в межах офісу та можуть бути представлені в GNS3.

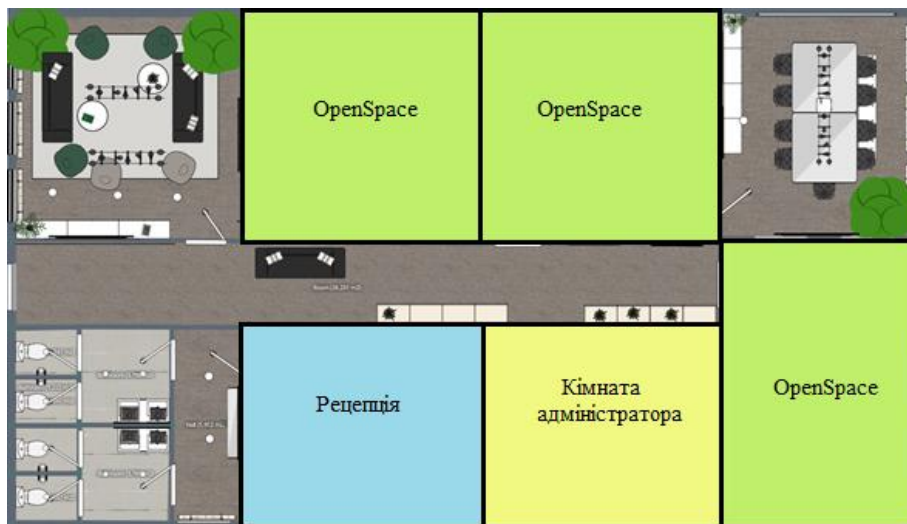


Рисунок 2.4 – Розподілення приміщення на зони

Мережа в GNS3 зпроектована для офісного приміщення з розрахунком, що OpenSpace має включати по 6 комп'ютерів для персоналу, а в кімнаті адміністратора знаходиться ядро мережі. Наведена кількість робочих місць є прикладом малого офісу. Для забезпечення відносної відмовостійкості та зменшення витрат на мережеве обладнання було створено схему:

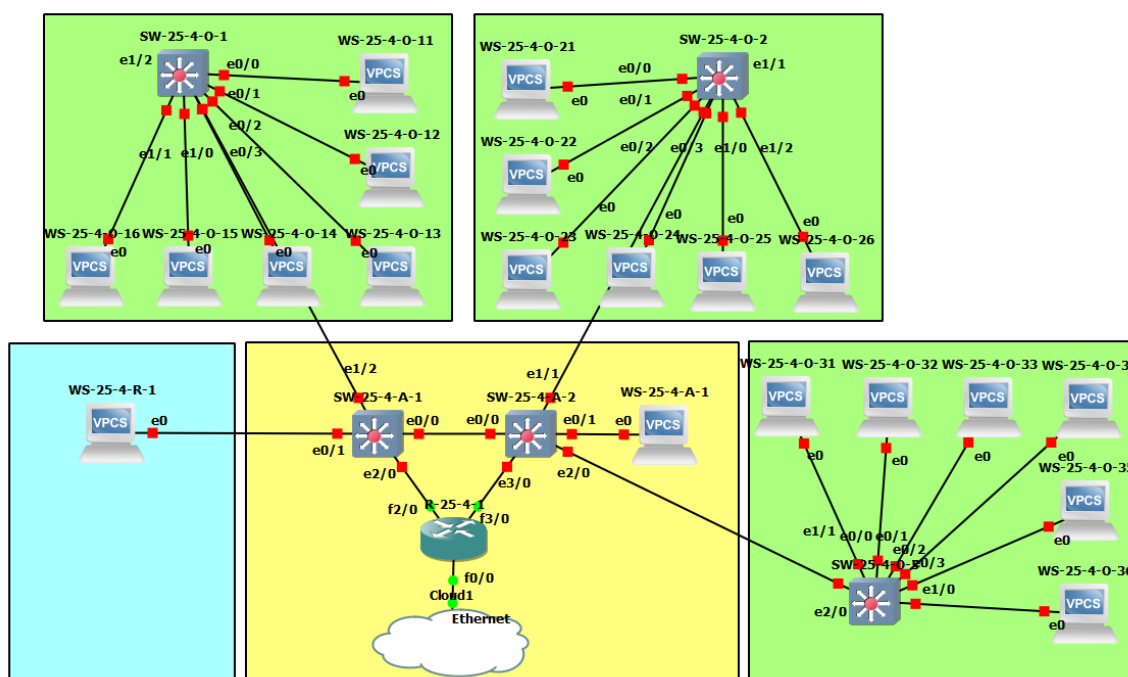


Рисунок 2.5 – Схема мережі

Розроблена схема адресації пристроїв мережі:

Таблиця 2.1

IP-адресація

Пристрій	Інтерфейс	IP-адреса	Маска підмережі	Підключено до пристрою	Інтерфейс	Префікс
R-25-4-1	Fa2/0	Bridge	255.255.255.224	SW-25-4-A-1	Eth2/0	27
	Fa3/0	194.25.4.1		SW-25-4-A-2	Eth3/0	
	Fa0/0	DHCP	255.255.0.0	Internet	-	24
SW-25-4-A-1	Vlan 1	194.25.4.2	255.255.255.224	SW-25-4-A-2	Eth0/0	27
				SW-25-4-O-1	Eth1/2	27
				WS-25-4-R-1	Eth0	27
				R-25-4-1	Fa2/0	27
SW-25-4-A-2	Vlan 1	194.25.4.3	255.255.255.224	SW-25-4-A-1	Eth0/0	27
				SW-25-4-O-2	Eth1/2	27
				SW-25-4-O-3	Eth2/0	27
				Serv-25-4-A-1	Eth0	27
				Serv-25-4-A-2	Eth0	27
				R-25-4-1	Fa3/0	27
SW-25-4-O-1	Vlan 1	194.25.4.4	255.255.255.224	WS-25-4-O-11	Eth0	27
				WS-25-4-O-12	Eth0	27
				WS-25-4-O-13	Eth0	27
				WS-25-4-O-14	Eth0	27
				WS-25-4-O-15	Eth0	27
				WS-25-4-O-16	Eth0	27
				SW-25-4-A-1	Fa1/2	27
SW-25-4-O-2	Vlan 1	194.25.4.5	255.255.255.224	WS-25-4-O-21	Eth0	27
				WS-25-4-O-22	Eth0	27
				WS-25-4-O-23	Eth0	27
				WS-25-4-O-24	Eth0	27
				WS-25-4-O-25	Eth0	27
				WS-25-4-O-26	Eth0	27
				SW-25-4-A-2	Eth1/2	27
SW-25-4-O-3	Vlan 1	194.25.4.6	255.255.255.224	WS-25-4-O-31	Eth0	27
				WS-25-4-O-32	Eth0	27
				WS-25-4-O-33	Eth0	27
				WS-25-4-O-34	Eth0	27
				WS-25-4-O-35	Eth0	27
				WS-25-4-O-36	Eth0	27
				SW-25-4-A-2	Eth2/0	27
WS-25-4-O-11	Eth0	DHCP	255.255.255.224	SW-25-4-O-1	Eth0/0	27
WS-25-4-O-12	Eth0				Eth0/1	27
WS-25-4-O-13	Eth0				Eth0/2	27
WS-25-4-O-14	Eth0				Eth0/3	27
WS-25-4-O-15	Eth0				Eth1/0	27
WS-25-4-O-16	Eth0				Eth1/1	27
WS-25-4-O-21	Eth0	DHCP	255.255.255.224	SW-25-4-O-2	Eth0/0	27
WS-25-4-O-22	Eth0				Eth0/1	27
WS-25-4-O-23	Eth0				Eth0/2	27
WS-25-4-O-24	Eth0				Eth0/3	27
WS-25-4-O-25	Eth0				Eth1/0	27
WS-25-4-O-26	Eth0				Eth1/1	27

Продовження таблиці 2.1

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

IP-адресація

Пристрій	Інтерфейс	IP-адреса	Маска підмережі	Підключено до пристрою	Інтерфейс	Префікс
WS-25-4-O-31	Eth0	DHCP	255.255.255.224	SW-25-4-O-3	Eth0/0	27
WS-25-4-O-32	Eth0				Eth0/1	27
WS-25-4-O-33	Eth0				Eth0/2	27
WS-25-4-O-34	Eth0				Eth0/3	27
WS-25-4-O-35	Eth0				Eth1/0	27
WS-25-4-O-36	Eth0				Eth1/1	27
WS-25-4-R-1	Eth0	194.25.4.28	255.255.255.224	SW-25-4-A-1	Eth0/1	27
Serv-25-4-A-1	Eth0	194.25.4.29	255.255.255.224	SW-25-4-A-2	Eth0/1	27
Serv-25-4-A-2	Eth0	194.25.4.30	255.255.255.224	SW-25-4-A-2	Eth0/1	27

Підмережі O, A, R відповідають за їхні належності до кабінетів: OpenSpace, кабінет адміністратора, рецепція відповідно. Сервер Serv-25-4-A-1 відповідає за моніторинг мережі, коли Serv-25-4-A-2 за керування доменом (див. рисунок А.1).

Домен — це унікальне ім'я, яке використовується для ідентифікації ресурсу в мережі "Інтернет" або в локальній мережі [5]. В LAN домен може позначати сукупність комп'ютерів, що об'єднані спільною системою адміністрування та політик безпеки (наприклад, у Windows-доменах на базі Active Directory). Домен полегшує організацію доступу до ресурсів, централізоване керування користувачами, пристроями та правами в мережі.

Налаштування мережі

```

R-25-4-1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R-25-4-1(config)#bridge irb
R-25-4-1(config)#bridge 1 protocol ieee
R-25-4-1(config)#int range fa2/0 , fa3/0
R-25-4-1(config-if-range)#bridge-group 1
R-25-4-1(config-if-range)#no ip address
R-25-4-1(config-if-range)#no sh
R-25-4-1(config-if-range)#int bvi 1
R-25-4-1(config-if)#ip address 190.25.3.1 255.255.255.252
R-25-4-1(config-if)#no sh
R-25-4-1(config-if)#int fa0/0
R-25-4-1(config-if)#ip add dhcp
R-25-4-1(config-if)#

```

Рисунок 2.6 – Приклад налаштування IP-адресації маршрутизатору

```

SW-25-4-A-1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
SW-25-4-A-1(config)#int vlan 1
SW-25-4-A-1(config-if)#ip address 194.25.4.2 255.255.255.224
SW-25-4-A-1(config-if)#no sh
SW-25-4-A-1(config-if)#exit
SW-25-4-A-1(config)#ip default-gateway 194.25.4.1
SW-25-4-A-1(config)#no ip domain-lookup
SW-25-4-A-1(config)#

```

Рисунок 2.7 – Приклад налаштування IP-адресації комутатору

При налаштуванні bridge-group стається помилка на мережевому рівні. Маршрутизатор не може пінгувати інші пристрої, в той час як його ip-адрес також недоступний. В такому випадку я налаштую тільки дротове з'єднання між комутатором SW-25-4-A-1 та роутером R-25-4-1.

Для забезпечення виходу локальної мережі в Інтернет на маршрутизаторі Cisco було налаштовано механізм NAT.

```

R-25-4-1(config)#access-list 1 permit 194.25.4.0 0.0.0.31
R-25-4-1(config)#int fa0/0
R-25-4-1(config-if)#ip nat outside
R-25-4-1(config-if)#int fa2/0
R-25-4-1(config-if)#ip nat inside
R-25-4-1(config-if)#ip nat inside source list 1 int fa0/0 overload

R-25-4-1(config)#
*Jun 13 14:16:25.703: %LINEPROTO-5-UPDOWN: Line protocol on Interface NVI0, changed state to up
R-25-4-1(config)#

```

Рисунок 2.8 – Налаштування NAT

NAT дозволяє транлювати внутрішні (приватні) IP-адреси локальної мережі у зовнішню (публічну) IP-адресу, яка використовується для доступу до глобальної мережі [6]. У рамках даного проєкту застосовано технологію PAT, що є різновидом NAT, при якому всі внутрішні пристрої використовують одну публічну IP-адресу, а розподіл сесій здійснюється за номером порту. Таке рішення дозволяє оптимізувати використання публічних IP-адрес та забезпечити коректну маршрутизацію запитів з внутрішньої мережі до Інтернету та у зворотному напрямку. Конфігурація NAT на маршрутизаторі Cisco включає визначення внутрішнього та зовнішнього інтерфейсів, створення списку доступу (ACL) для внутрішньої адресації та застосування відповідної команди ip nat inside source list для активації трансляції [7].

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Для автоматичної роздачі IP-адрес у локальній мережі була налаштована служба DHCP на маршрутизаторі Cisco для зони LAN. У рамках конфігурації було визначено пул доступних адрес для динамічної видачі робочим станціям та іншим клієнтам мережі. Для ключових мережевих пристроїв, таких як комутатори та сервери, було створено виключення (DHCP exclusions), щоб ці пристрої мали статичні IP-адреси, необхідні для стабільної роботи мережевих служб та управління. Крім того, у налаштування DHCP-пулу були включені параметри DNS-сервера для забезпечення можливості доменних запитів клієнтами, а також вказано значення доменного імені (domain-name), що використовується у локальній мережі для централізованого управління та автентифікації в доменній структурі. Це дозволяє забезпечити коректну інтеграцію клієнтських пристроїв до домену та спростити адміністрування мережі.

```
R-25-4-1(config)#ip dhcp excluded-address 194.25.4.1 194.25.4.2
R-25-4-1(config)#ip dhcp excluded-address 194.25.4.3 194.25.4.4
R-25-4-1(config)#ip dhcp excluded-address 194.25.4.5 194.25.4.6
R-25-4-1(config)#ip dhcp excluded-address 194.25.4.28 194.25.4.29
R-25-4-1(config)#ip dhcp excluded-address 194.25.4.30
R-25-4-1(config)#ip dhcp pool LAN
R-25-4-1(dhcp-config)#default-router 194.25.4.1
R-25-4-1(dhcp-config)#dns-server 194.25.4.1
R-25-4-1(dhcp-config)#domain-name zmiievskyi.net
R-25-4-1(dhcp-config)#exit
R-25-4-1(config)#
```

Рисунок 2.9 – Налаштування DHCP

2.3 Вибір мови програмування для розробки програмного агента

Програмний агент для контролю доступу може складатися з декількох модулів і забезпечувати моніторинг різних видів ресурсів у мережі, таких як використання процесора, оперативної пам'яті, дискових накопичувачів або мережевого трафіку [8]. Незважаючи на можливість використання кількох мов програмування для різних частин системи, з метою спрощення розробки та подальшого супроводу проєкту прийнято рішення використовувати єдину мову програмування для всіх компонентів агента. Вибір конкретної мови базується на попередньому аналізі доступних технологій, врахуванні специфіки задачі, вимог до продуктивності, можливостей взаємодії з мережею та сумісності із середовищем розгортання програмного засобу [9].

Порівняльна характеристика мов програмування

Критерій	Python	C/C++	Go (Golang)
Простота розробки	Висока. Зручний синтаксис, велика кількість бібліотек.	Низька. Складний синтаксис, ручне управління пам'яттю.	Середня. Простий синтаксис, створено для системних застосунків.
Продуктивність (швидкість виконання)	Середня. Інтерпретується, повільніше за C/C++.	Висока. Машинний код, максимальна швидкість.	Висока. Компільована мова, продуктивність наближена до C/C++.
Управління пам'яттю	Автоматичне (збір сміття).	Ручне. Потрібно явно виділяти та звільняти пам'ять.	Автоматичне (вбудований збирач сміття).
Мережеві можливості	Відмінні. Велика кількість бібліотек для роботи з мережами (наприклад, socket, requests, scrapy).	Потрібні сторонні бібліотеки або складна реалізація.	Добрі. Вбудовані засоби для роботи з HTTP, TCP, UDP.
Паралелізм/многопоточність	Слабка через GIL (Global Interpreter Lock).	Висока, залежить від ОС та реалізації.	Відмінна. Простий паралелізм за допомогою "горутин".
Кросплатформеність	Відмінна. Працює майже на всіх платформах.	Висока, але потребує перекомпіляції.	Відмінна. Має вбудований кроскомпілятор.
Застосування	Сценарії, автоматизація, мережеві агенти, ML.	Системне програмування, драйвери, високопродуктивні системи.	Серверні застосунки, мережеві сервіси, мікросервіси.
Порог входу	Низький — легко вивчити початківцям.	Високий — потрібне розуміння системного рівня.	Середній — спрощений синтаксис, але нова концепція горутин.

Ураховуючи специфіку завдання, вимоги до швидкої розробки та підтримки, доступність бібліотек і зручність роботи з мережею, мовою реалізації програмного агента було обрано Python.

2.4 Вибір мережевих протоколів для обміну даними

У процесі розробки програмного агента для контролю доступу до ресурсів та виявлення аномальної активності у локальній мережі виникає потреба у використанні ефективних та безпечних мережевих протоколів для обміну інформацією між компонентами системи. Основні вимоги до вибору протоколів включають: надійність передачі даних, забезпечення конфіденційності, цілісності інформації, сумісність із програмним середовищем та простоту реалізації.

Для передачі даних у системі обрано такі протоколи:

HTTP/HTTPS

Протокол HTTP (HyperText Transfer Protocol) використовується для надсилання запитів та отримання відповідей від різних компонентів системи, зокрема для взаємодії програмного агента з сервером моніторингу (наприклад Grafana). Враховуючи вимоги до захисту переданої інформації від можливих атак (перехоплення або модифікації даних), доцільно використовувати протокол HTTPS — захищену версію HTTP, яка застосовує шифрування TLS/SSL.

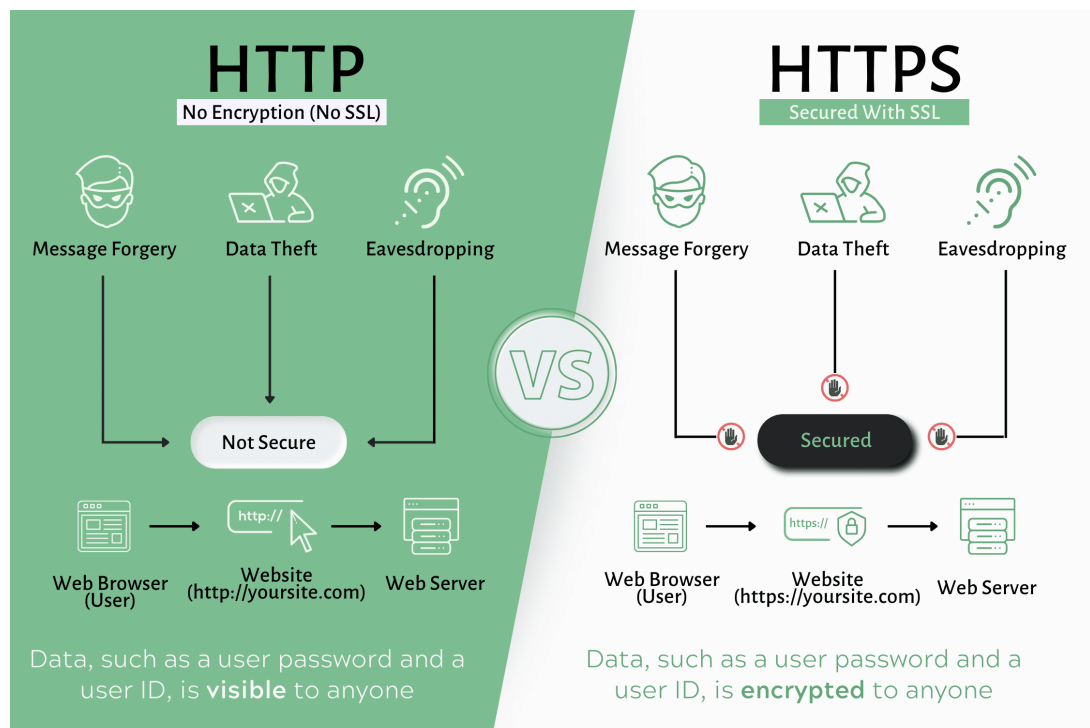


Рисунок 2.10 – Різниця між HTTP, HTTPS [10]

Використання HTTPS забезпечує:

- захист від перехоплення даних;
- автентифікацію сервера;
- запобігання модифікації трафіку сторонніми особами.

Таким чином, запити на отримання або відправлення метрик, а також взаємодія з Grafana Dashboard здійснюватимуться через HTTPS.

TCP/IP

Базовим протоколом транспортного та мережевого рівнів обрано стек TCP/IP, який є основою для побудови сучасних комп'ютерних мереж. Зокрема, протокол TCP (Transmission Control Protocol) забезпечує надійну доставку даних між агентом та сервером моніторингу з контролем цілісності інформації, порядку передавання пакетів та коректного завершення сеансів зв'язку [11].

Переваги TCP:

- гарантія доставки пакетів;
- контроль помилок та відновлення втраченої інформації;
- забезпечення послідовності даних.

Для внутрішніх обмінів у межах локальної мережі також застосовується протокол IP для маршрутизації пакетів.

Telegram Bot API (HTTPS)

Для реалізації сповіщень про аномальну активність або важливі події використовується Telegram Bot API, який працює поверх HTTPS. Це дозволяє безпечно та надійно передавати повідомлення від програмного агента до кінцевого користувача [12].

Telegram Bot API надає:

- зручний інтерфейс для відправлення текстових сповіщень;
- підтримку асинхронних запитів;
- захищене з'єднання через HTTPS.

Висновки до другого розділу

У даному розділі було виконано комплексне проєктування інфраструктури для розробки програмного агента, що здійснює моніторинг ресурсів локальної мережі та контроль доступу до них.

На першому етапі було розроблено план офісного приміщення, який включає різноманітні функціональні зони, що дозволяє оптимально розмістити технічні та інформаційні ресурси підприємства.

Наступним кроком була побудова моделі комп'ютерної мережі у середовищі GNS3 (див. рисунок Б.1). У моделі реалізовано логічне структурування підмереж для різних зон офісу, створено IP-адресацію, визначено маршрутизацію між сегментами мережі, налаштовано комутатори та маршрутизатори. Для централізованого управління доступом створено домен, що дозволяє спростити автентифікацію користувачів та адміністрування мережі.

У підрозділі з вибору мови програмування було проведено порівняння популярних мов — Python, C/C++ та Go, — на основі якого обґрунтовано вибір мови Python для реалізації програмного агента. Python забезпечує простоту розробки, наявність великої кількості готових бібліотек для роботи з мережами, кросплатформеність та можливість швидкої інтеграції з такими системами, як Grafana та Telegram API.

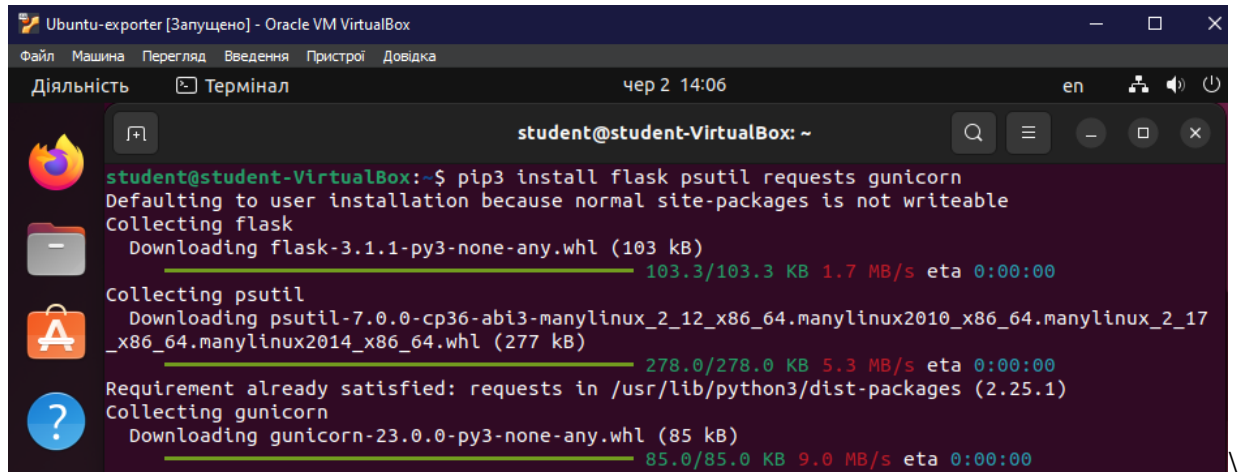
Для обміну даними в системі обрано перевірені та надійні протоколи — транспортного і прикладного рівня, TCP/IP та HTTPS відповідно, що забезпечують захист переданої інформації та сумісність з основними компонентами системи (Grafana, Telegram API). Такий вибір дозволяє досягти високої надійності та безпеки роботи програмного агента у локальній мережі.

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСОБУ

3.1 Архітектура програмного агента

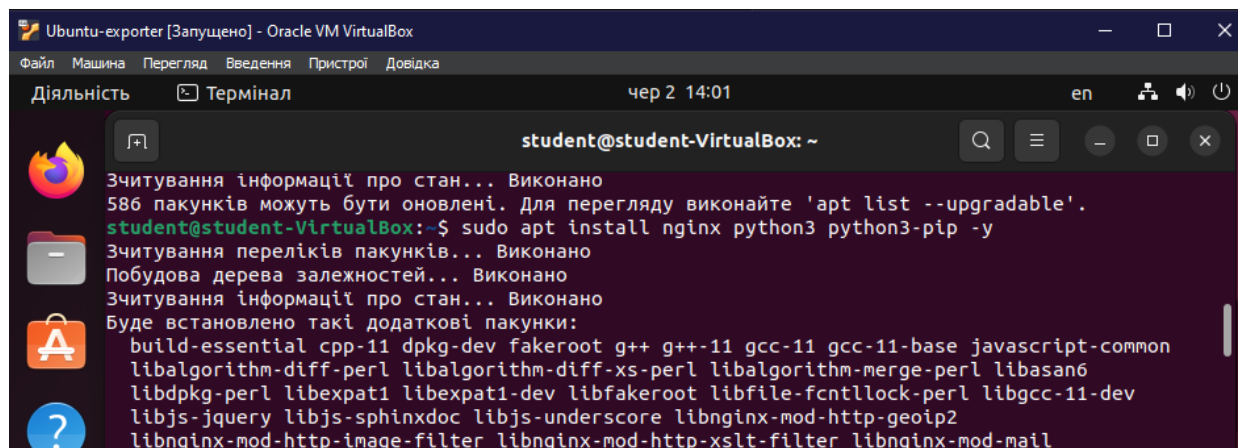
У складі програмного агента моніторингу важливу роль відіграють компоненти Nginx та Flask, які забезпечують взаємодію між агентом, серверною частиною та зовнішніми сервісами.



```
student@student-VirtualBox: ~  
student@student-VirtualBox:~$ pip3 install flask psutil requests gunicorn  
Defaulting to user installation because normal site-packages is not writeable  
Collecting flask  
  Downloading flask-3.1.1-py3-none-any.whl (103 kB)  
    103.3/103.3 KB 1.7 MB/s eta 0:00:00  
Collecting psutil  
  Downloading psutil-7.0.0-cp36-abi3-manylinux_2_12_x86_64.manylinux2010_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (277 kB)  
    278.0/278.0 KB 5.3 MB/s eta 0:00:00  
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (2.25.1)  
Collecting gunicorn  
  Downloading gunicorn-23.0.0-py3-none-any.whl (85 kB)  
    85.0/85.0 KB 9.0 MB/s eta 0:00:00
```

Рисунок 3.1 – Встановлення flask

Flask виконує роль веб-сервера застосунку та API. За допомогою Flask реалізовано HTTP(S)-ендпоінти для прийому метрик від клієнтського агента, а також для надання цих даних інструментам візуалізації, таким як Grafana, через JSON API. Flask обробляє запити, виконує запис отриманих даних у базу SQLite та формує відповіді для клієнтів або зовнішніх систем [13][14]. Гнучкість Flask дозволяє легко розширювати API, додавати нові маршрути або логіку обробки даних.



```
student@student-VirtualBox: ~  
student@student-VirtualBox:~$ sudo apt install nginx python3 python3-pip -y  
Зчитування інформації про стан... Виконано  
586 пакунків можуть бути оновлені. Для перегляду виконайте 'apt list --upgradable'.  
Зчитування переліків пакунків... Виконано  
Побудова дерева залежностей... Виконано  
Зчитування інформації про стан... Виконано  
Буде встановлено такі додаткові пакунки:  
  build-essential cpp-11 dpkg-dev fakeroot g++ g++-11 gcc-11 gcc-11-base javascript-common  
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6  
  libdpkg-perl libexpat1 libexpat1-dev libfakeroot libfile-fcntllock-perl libgcc-11-dev  
  libjs-jquery libjs-sphinxdoc libjs-underscore libnginx-mod-http-geoip2  
  libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod-mail
```

Рисунок 3.2 – Встановлення nginx

Nginx виконує роль зворотного проксі-сервера (reverse proxy) для Flask-додатка. Він приймає всі зовнішні HTTP(S)-запити, виконує SSL-термінацію (розшифровку HTTPS-з'єднання із застосуванням самопідписаного SSL-сертифіката) та пересилає запити до локального Flask-сервера, який працює через Gunicorn [16]. Завдяки цьому забезпечується захист переданих даних, можливість балансування навантаження (у разі масштабування системи) та зручна маршрутизація запитів до різних сервісів [15].

```
GNU nano 6.2 /etc/nginx/sites-available/monitor
server {
    listen 443 ssl;
    server_name localhost;

    ssl_certificate /etc/ssl/mycerts/selfsigned.crt;
    ssl_certificate_key /etc/ssl/mycerts/selfsigned.key;

    location / {
        proxy_pass http://127.0.0.1:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}

server {
    listen 80;
    server_name localhost;

    return 301 https://$host$request_uri;
}
```

Рисунок 3.3 – Редагування конфігураційного файлу nginx

Для файлу `/etc/nginx/sites-available/monitor` були внесені зміни для забезпечення безпечного доступу до веб-застосунку через протокол HTTPS. Зокрема, вказано, що сервер слухає порт 443 із використанням SSL-сертифіката та закритого ключа, розташованих у каталозі `/etc/ssl/mycerts/`. Усі вхідні запити проксуються на внутрішній Flask-сервер, який працює на локальному інтерфейсі `127.0.0.1:5000`. Додатково налаштовано редирект із порту 80 на HTTPS, щоб усі запити, які надходять через незахищений HTTP, автоматично перенаправлялися на захищене з'єднання.

```
student@student-VirtualBox:~$ sudo nano /etc/nginx/sites-available/monitor
[sudo] пароль до student:
student@student-VirtualBox:~$ sudo ln -s /etc/nginx/sites-available/monitor /etc/nginx/sites-enabled/
student@student-VirtualBox:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
student@student-VirtualBox:~$ sudo systemctl restart nginx
student@student-VirtualBox:~$
```

Рисунок 3.4 – Налаштування nginx

У Nginx конфігураційні файли для доступних (але ще не активованих) сайтів зберігаються в `/etc/nginx/sites-available/`, а для активних — у `/etc/nginx/sites-enabled/`. Створюючи символічне посилання, активується новий конфігураційний файл без необхідності дублювати його, що дозволяє зручно керувати сайтами (вмикати/вимикати їх за допомогою посилань).

Програмний агент контролю доступу до ресурсів, реалізовано в середовищі розробки PyCharm. Розробка почалася з базової структури для обробки HTTP-запитів, що дозволяє перевірити працездатність сервера та приймати метрики від клієнтських агентів у форматі JSON. Код програми зберігається у локальному репозиторії Git, що дозволяє вести контроль версій розробки, а також забезпечує можливість подальшого завантаження (push) змін до віддаленого репозиторію на платформі GitLab для централізованого зберігання, спільної роботи та забезпечення безпеки програмного коду [17].

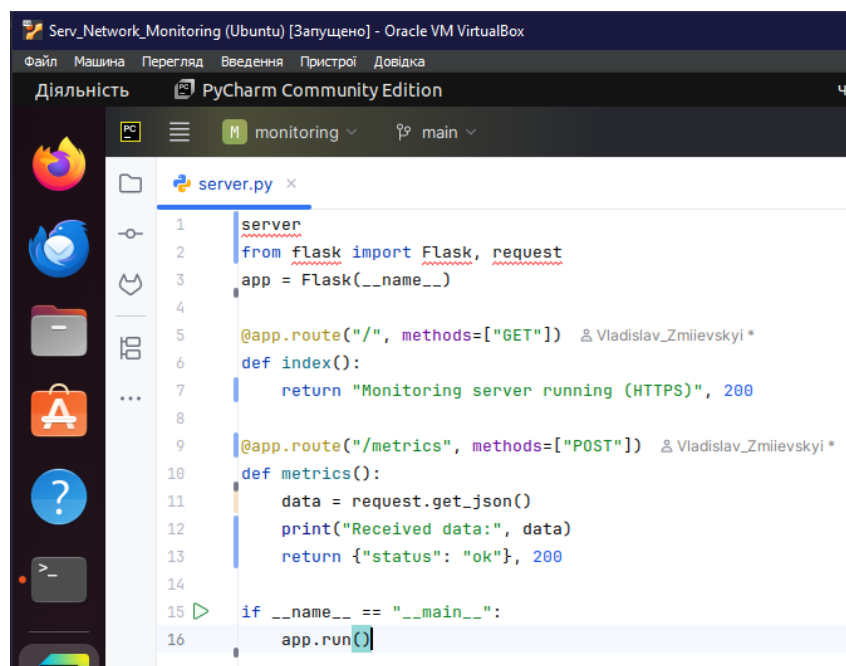


Рисунок 3.5 – Написаний код для прийняття метрик

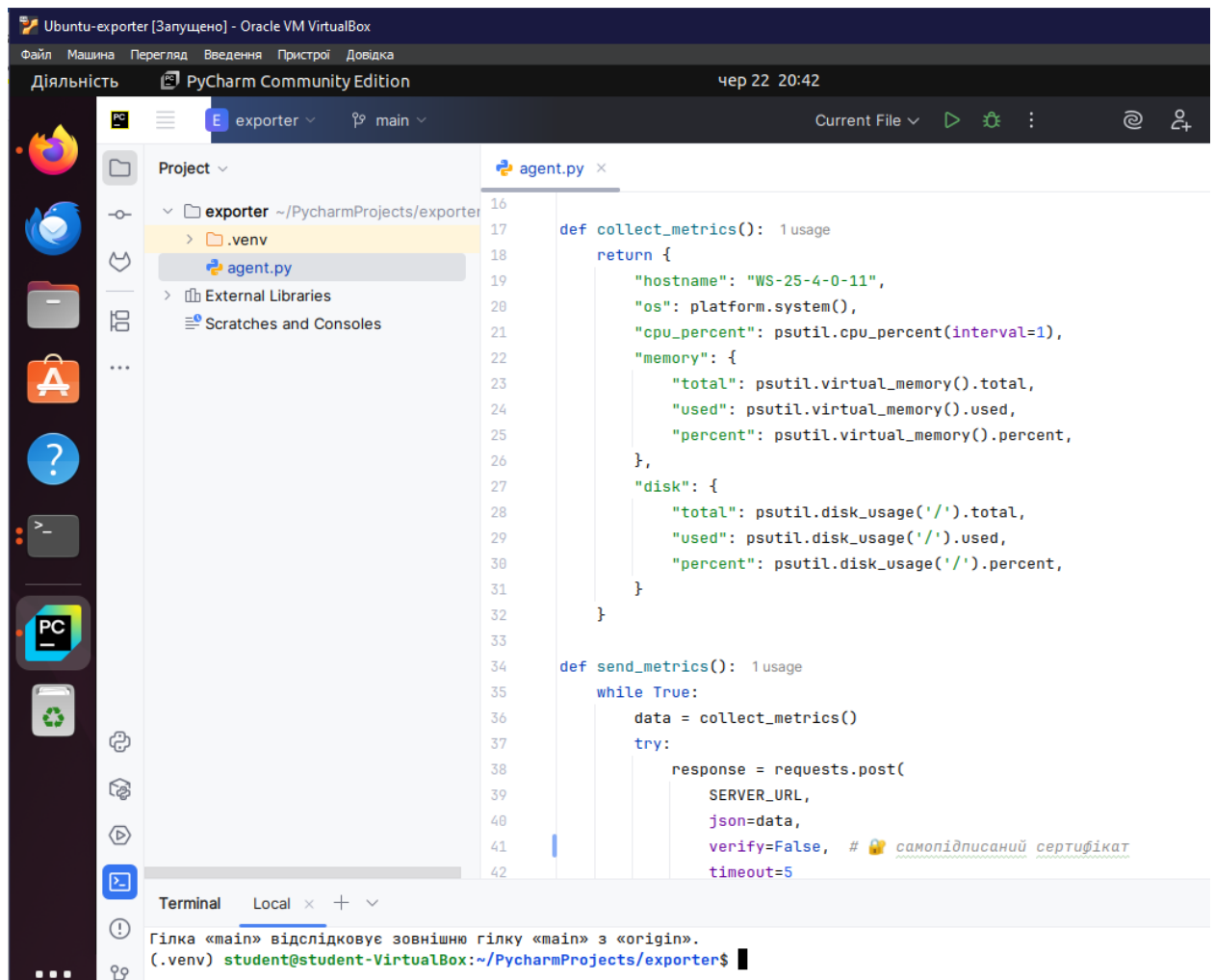


Рисунок 3.6 – Налаштування клієнтського застосунку

Виконуваний файл `agent.py` виконує роль клієнтського застосунку, який встановлюється на хості, що підлягає моніторингу, та збирає інформацію про стан системних ресурсів, таких як завантаження процесора (CPU), використання оперативної пам'яті (RAM) та заповненість дискового простору. Отримані метрики обробляються локально, після чого агент формує структуру даних у форматі JSON та надсилає її на сервер моніторингу через захищене HTTPS-з'єднання до спеціального ендпоінту Flask-додатку. Робота агента організована у вигляді циклу із заданим інтервалом часу, що дозволяє здійснювати безперервний контроль за станом системи та оперативно реагувати на виявлені аномалії або перевантаження ресурсів.

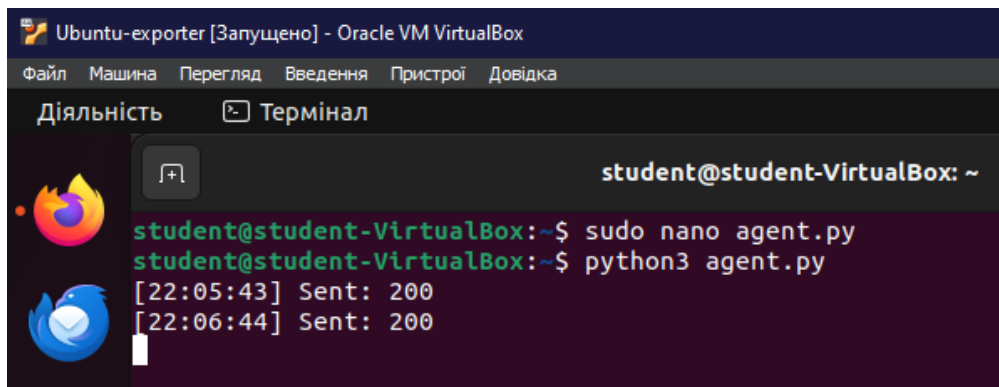


Рисунок 3.7 – Відправлення метрик на сервер

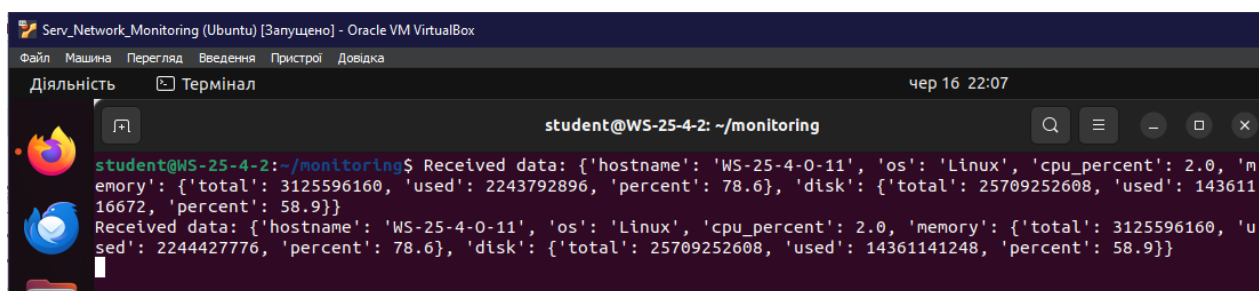


Рисунок 3.8 – Отримання метрик сервером

3.2 Опис REST API-запитів до сервера моніторингу

У цьому підрозділі розглядаються HTTP-запити, які обробляє сервер моніторингу [18]. Сервер реалізує REST API, що дозволяє клієнтам (агентам моніторингу) надсилати дані про стан системи та отримувати останні записані метрики [19]. Найважливішими є методи POST (для надсилання метрик) та GET (для отримання останніх записів з бази даних).

Кожен метод прив'язаний до певного маршруту (endpoint) і працює відповідно до загальноприйнятих принципів REST-архітектури.

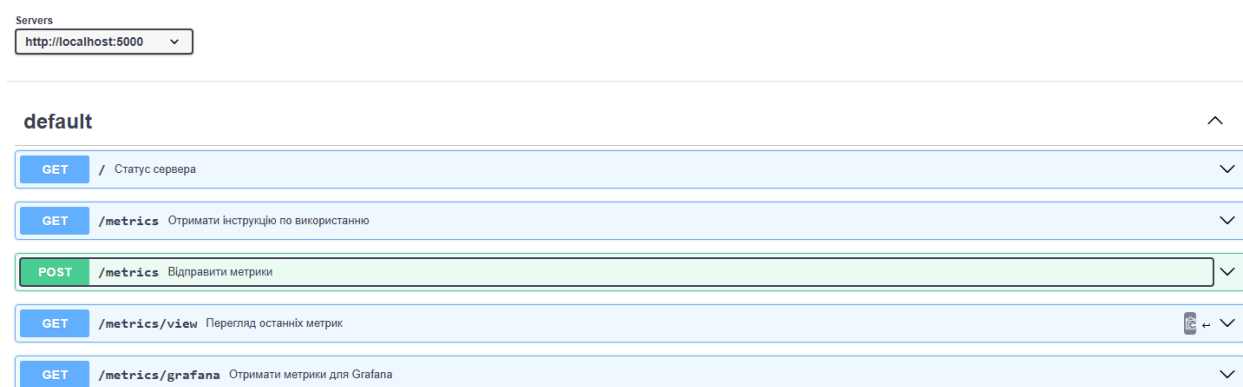


Рисунок 3.9 – Реалізовані методи

Метод GET / повертає текст «Monitoring server is running.», якщо сервер працює. Перевірка доступності сервера.

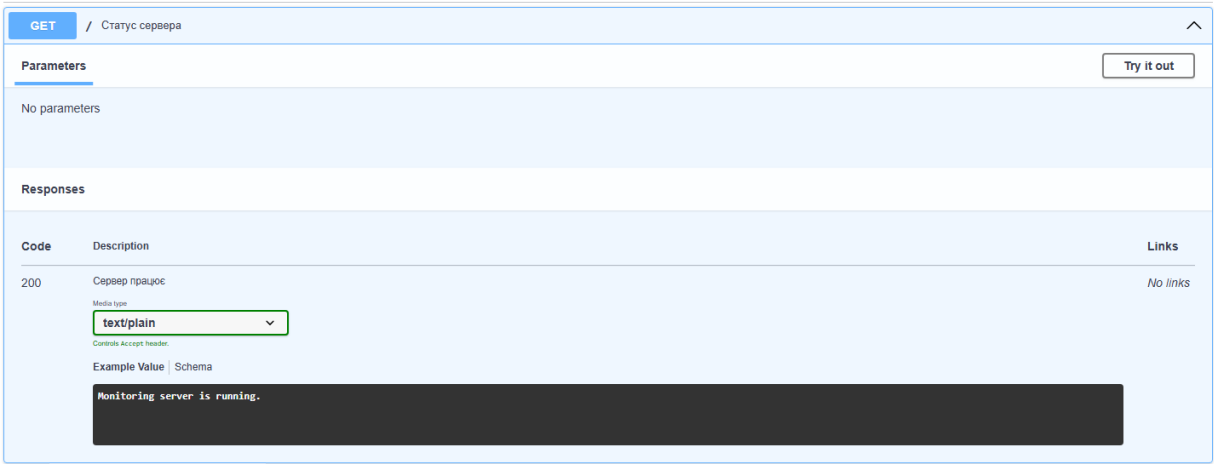


Рисунок 3.10 – Метод GET /

Метод POST /metrics приймає JSON з метриками (CPU, RAM, диск):



Рисунок 3.11 – Метод POST / metrics

В цьому методі відбувається перевірка обов’язкових полів (hostname, cpi_percent тощо). Також він відповідає за систему оповіщень, якщо передані метрики перевищують певні пороги (наприклад, cpi_percent > 90). Сервер формує відповідне повідомлення та надсилає його адміністратору через Telegram-бота.

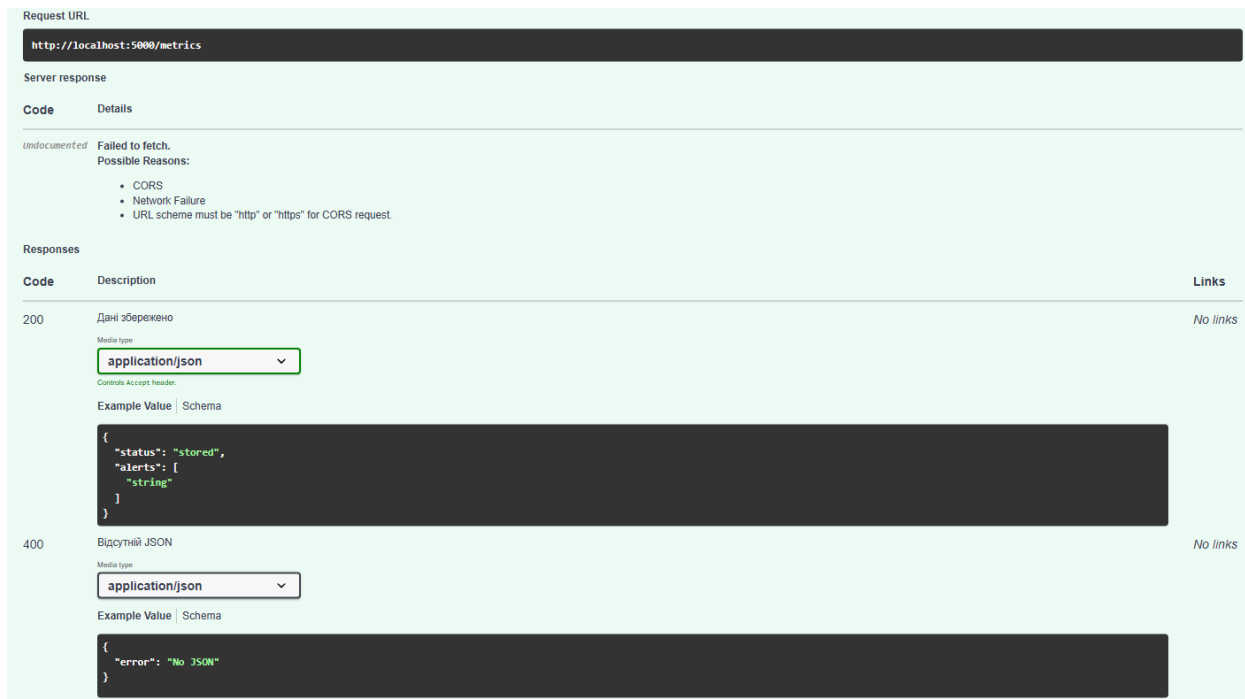


Рисунок 3.12 – Метод POST / metrics

Метод POST використовується для надсилання метрик, тому GET /metrics просто повертає текстову або JSON-відповідь з описом; інформує користувача або клієнта про доступність цього endpoint і правила його використання [20].

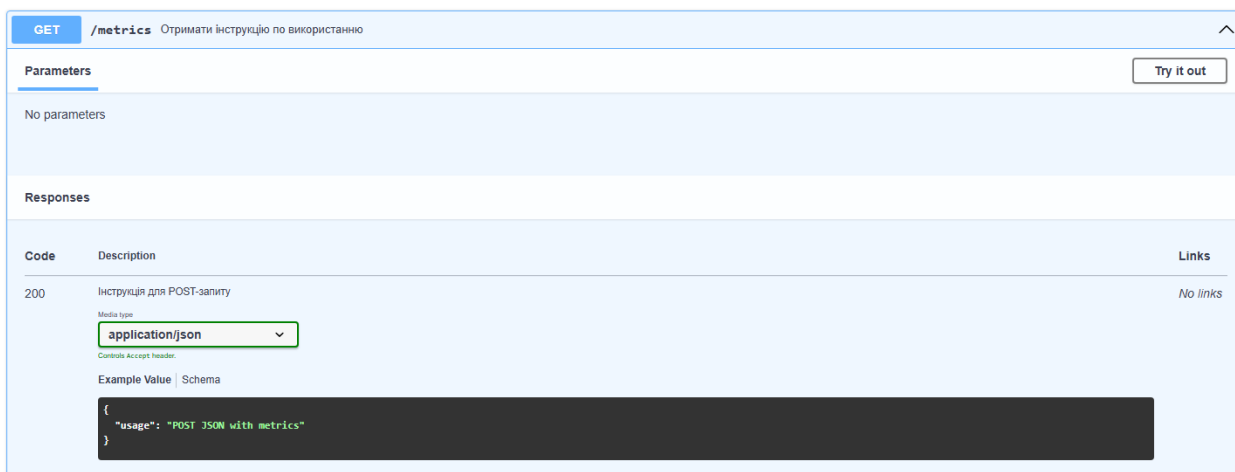


Рисунок 3.13 – Метод GET /metrics

Даний метод корисний для відлагодження або перегляду API без помилок типу 405 Method Not Allowed.

GET /metrics/view повертає останні збережені метрики з бази даних у форматі JSON. Цей endpoint використовується для перегляду даних вручну:

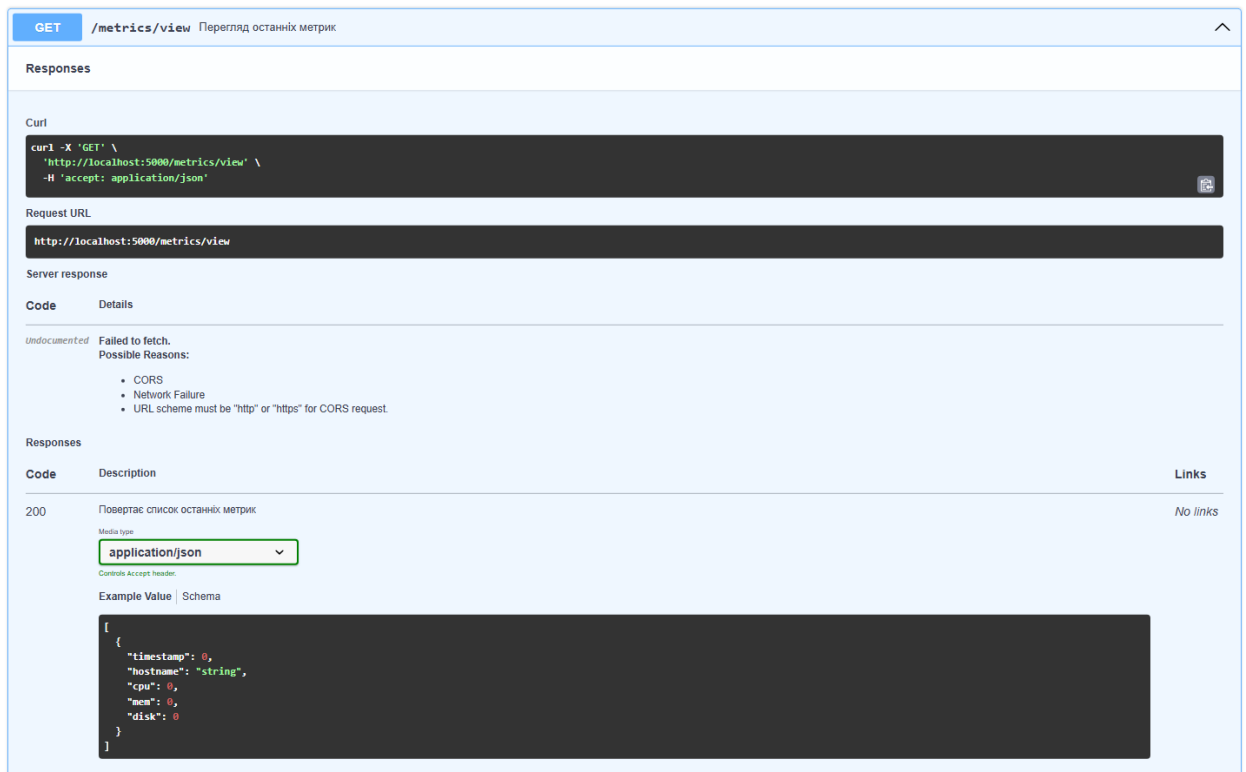


Рисунок 3.14 – Метод GET /metrics/view

Для отримання метрик сумісних з Grafana JSON API datasource було створено метод GET /metrics/grafana. Цей endpoint створений спеціально для побудови графіків та візуалізації даних у реальному часі:

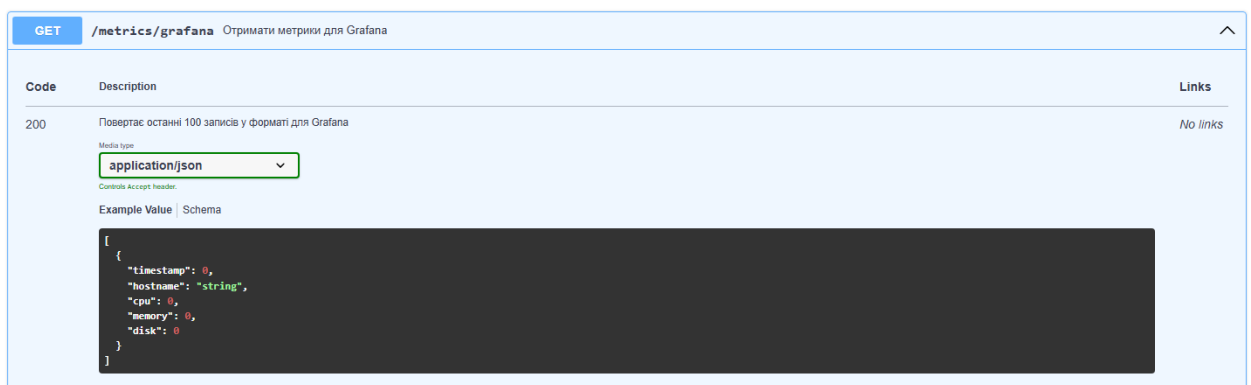


Рисунок 3.15 – Метод GET / metrics/grafana

3.3 Виявлення аномальної активності

Для підвищення надійності та оперативності реагування система моніторингу реалізує механізм виявлення аномальної активності, який автоматично аналізує отримані метрики від агентів. У разі перевищення критичних порогів, таких як надмірне використання процесора або

оперативної пам'яті, система формує сповіщення і відправляє його адміністратору через Telegram-бота.

Telegram було обрано як основний канал оповіщення з огляду на такі переваги:

Таблиця 3.1

Переваги вибору Telegram для оповіщення

Критерій	Пояснення
Швидкість доставки	Повідомлення надходять миттєво, незалежно від типу пристрою
Кросплатформеність	Telegram доступний на Windows, Linux, Android, iOS, Web
Простота API	Telegram надає зручний бот-інтерфейс для надсилання повідомлень через HTTPS
Безпека	Підтримка HTTPS-з'єднання та внутрішнє шифрування
Відсутність реклами та обмежень	Telegram не накладає ліміти на кількість повідомлень, як деякі інші сервіси
Легка реалізація	Надсилання повідомлень реалізується за допомогою простого HTTP POST-запиту без додаткових бібліотек або ключів доступу до сторонніх платформ

Для інтеграції telegram-боту потрібно його створити через @BotFather (див. рисунок Д.1), після чого ми отримуємо його токен. Також потрібно ID чату, яке можна отримати на сторінці <https://api.telegram.org> (див. рисунок Д.2).

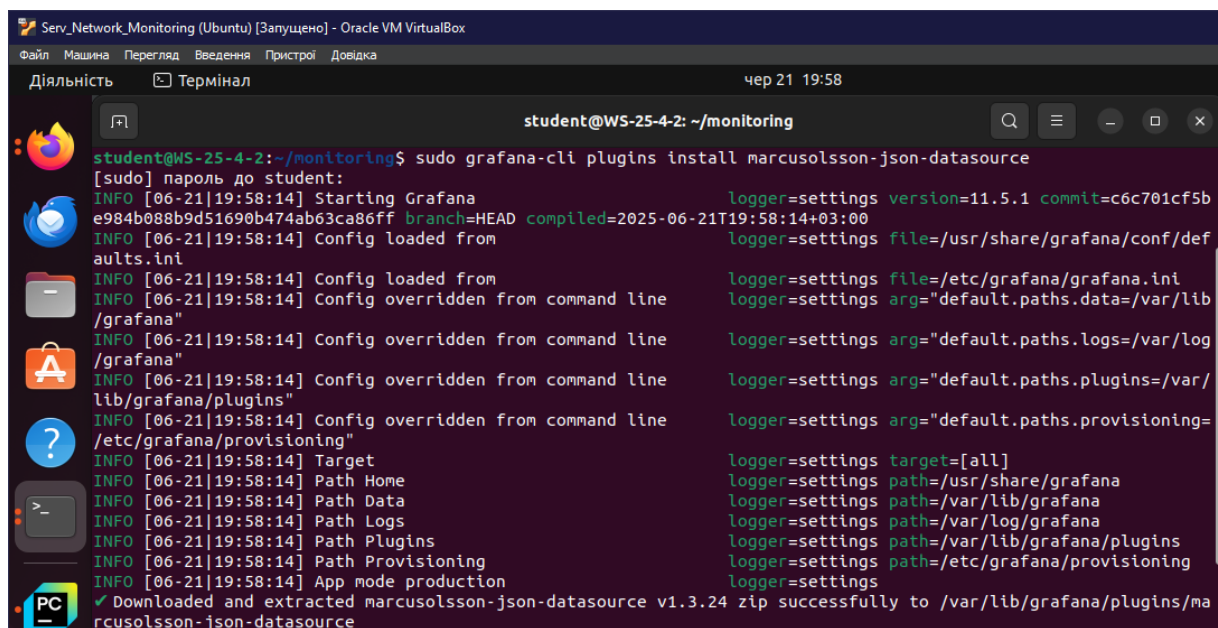
Інтеграція Telegram потребує встановлення бібліотеки:

```
(.venv) student@WS-25-4-2:~/monitoring$ pip3 install requests
Collecting requests
  Downloading requests-2.32.4-py3-none-any.whl.metadata (4.9 kB)
Collecting charset_normalizer<4,>=2 (from requests)
  Downloading charset_normalizer-3.4.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (35 kB)
Collecting idna<4,>=2.5 (from requests)
  Downloading idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting urllib3<3,>=1.21.1 (from requests)
  Downloading urllib3-2.4.0-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi>=2017.4.17 (from requests)
  Downloading certifi-2025.6.15-py3-none-any.whl.metadata (2.4 kB)
Downloading requests-2.32.4-py3-none-any.whl (64 kB)
Downloading charset_normalizer-3.4.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (149 kB)
Downloading idna-3.10-py3-none-any.whl (70 kB)
Downloading urllib3-2.4.0-py3-none-any.whl (128 kB)
Downloading certifi-2025.6.15-py3-none-any.whl (157 kB)
Installing collected packages: urllib3, idna, charset_normalizer, certifi, requests
Successfully installed certifi-2025.6.15 charset_normalizer-3.4.2 idna-3.10 requests-2.32.4 urllib3-2.4.0
(.venv) student@WS-25-4-2:~/monitoring$
```

Рисунок 3.16 – Встановлення requests

3.4 Візуалізація та представлення даних у Grafana

Для зручного аналізу зібраних метрик у проєкті використовується система візуалізації Grafana, яка дозволяє наочно відображати дані про стан ресурсів моніторингових хостів. Завдяки використанню плагіна JSON API у Grafana здійснюється отримання актуальних значень метрик безпосередньо з Flask-сервера через спеціально реалізований HTTP(S)-ендпоінт /metrics/grafana [21].



```
student@WS-25-4-2: ~/monitoring
student@WS-25-4-2:~/monitoring$ sudo grafana-cli plugins install marcusolsson-json-datasource
[sudo] пароль до student:
INFO [06-21|19:58:14] Starting Grafana                      logger=settings version=11.5.1 commit=c6c701cf5b
e984b088b9d51690b474ab63ca86ff branch=HEAD compiled=2025-06-21T19:58:14+03:00
INFO [06-21|19:58:14] Config loaded from                          logger=settings file=/usr/share/grafana/conf/def
aults.ini
INFO [06-21|19:58:14] Config loaded from                          logger=settings file=/etc/grafana/grafana.ini
INFO [06-21|19:58:14] Config overridden from command line          logger=settings arg="default.paths.data=/var/lib
/grafana"
INFO [06-21|19:58:14] Config overridden from command line          logger=settings arg="default.paths.logs=/var/log
/grafana"
INFO [06-21|19:58:14] Config overridden from command line          logger=settings arg="default.paths.plugins=/var/
lib/grafana/plugins"
INFO [06-21|19:58:14] Config overridden from command line          logger=settings arg="default.paths.provisioning=
/etc/grafana/provisioning"
INFO [06-21|19:58:14] Target                                      logger=settings target=[all]
INFO [06-21|19:58:14] Path Home                                  logger=settings path=/usr/share/grafana
INFO [06-21|19:58:14] Path Data                                  logger=settings path=/var/lib/grafana
INFO [06-21|19:58:14] Path Logs                                  logger=settings path=/var/log/grafana
INFO [06-21|19:58:14] Path Plugins                               logger=settings path=/var/lib/grafana/plugins
INFO [06-21|19:58:14] Path Provisioning                         logger=settings path=/etc/grafana/provisioning
INFO [06-21|19:58:14] App mode production                       logger=settings
✓ Downloaded and extracted marcusolsson-json-datasource v1.3.24 zip successfully to /var/lib/grafana/plugins/mar
cusolsson-json-datasource
```

Рисунок 3.19 – Встановлення Grafana

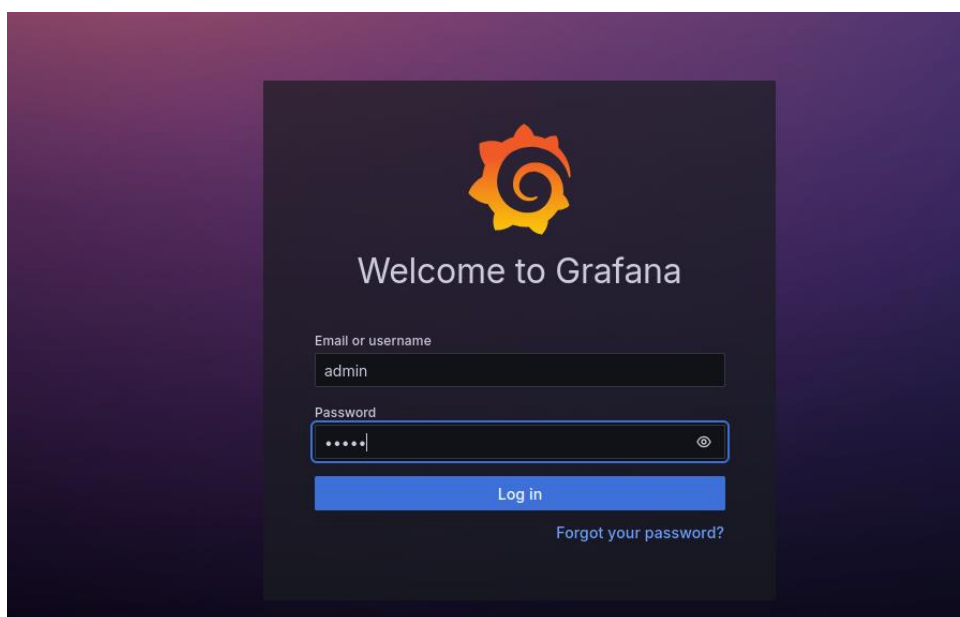


Рисунок 3.20 – Авторизація в Grafana

Для забезпечення доступу Grafana до метрик, зібраних програмним агентом, було налаштовано спеціальне джерело даних (Data Source) з використанням плагіна JSON API:

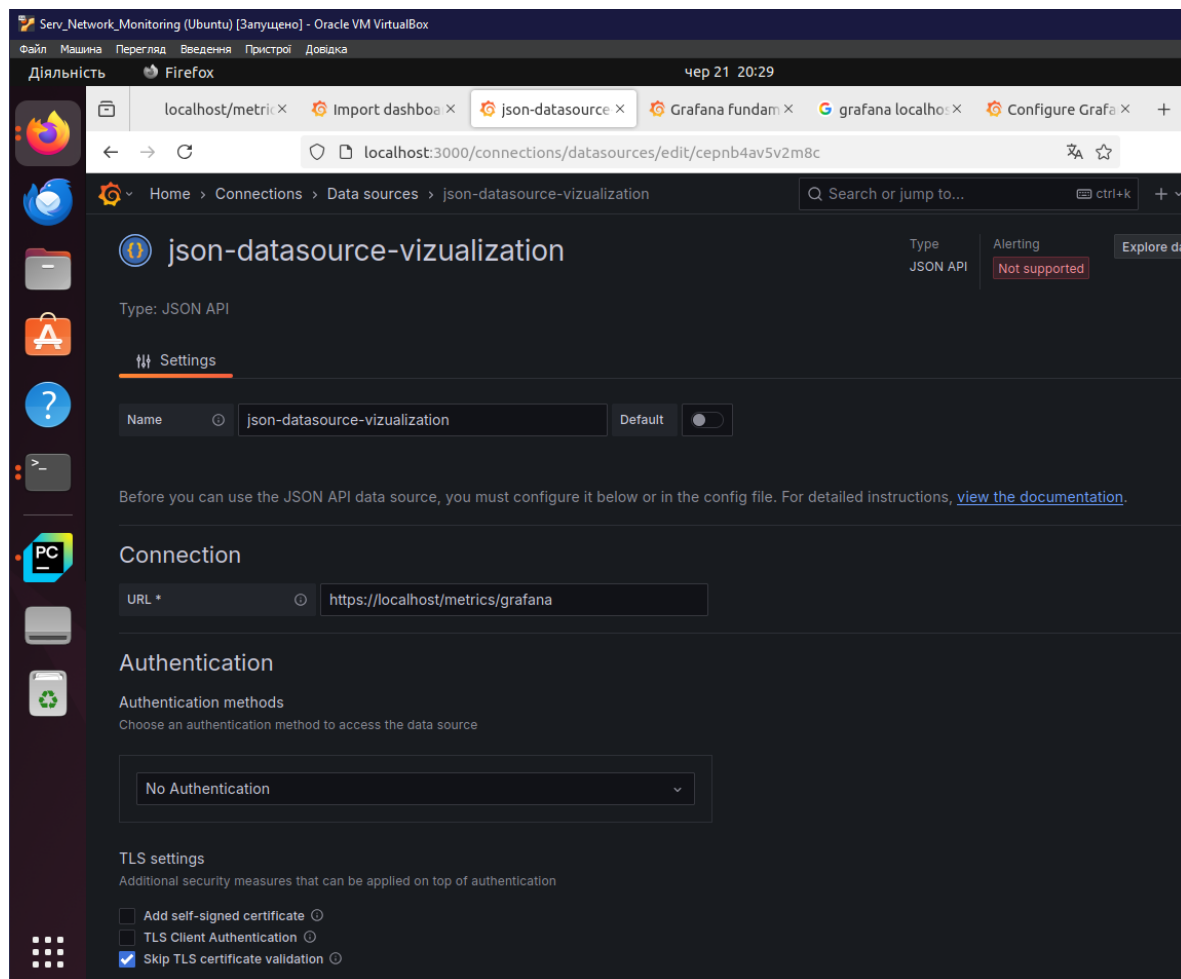


Рисунок 3.21 – Створення джерела даних

В поле URL введено адресу сервера, на якому працює Flask-додаток: `https://localhost/metrics/grafana`. Це дозволяє Grafana періодично надсилати запити до даного ендпоінту та отримувати оновлені значення метрик у форматі JSON. Для забезпечення захищеного з'єднання використовується протокол HTTPS із самопідписаним SSL-сертифікатом, налаштованим на веб-сервері Nginx.

Такий підхід дозволяє здійснювати інтеграцію власного програмного рішення з Grafana без використання сторонніх систем моніторингу, таких як Prometheus або Zabbix, забезпечуючи повну контрольованість процесу обробки та представлення метрик.

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

Імпорт готового dashboard у Grafana за допомогою JSON-моделі дозволяє швидко відновити або розгорнути заздалегідь налаштовану панель візуалізації:

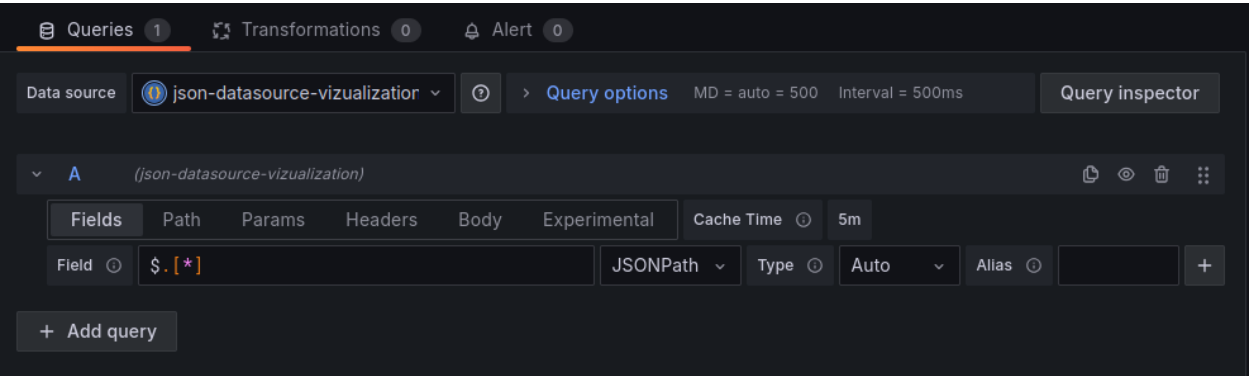
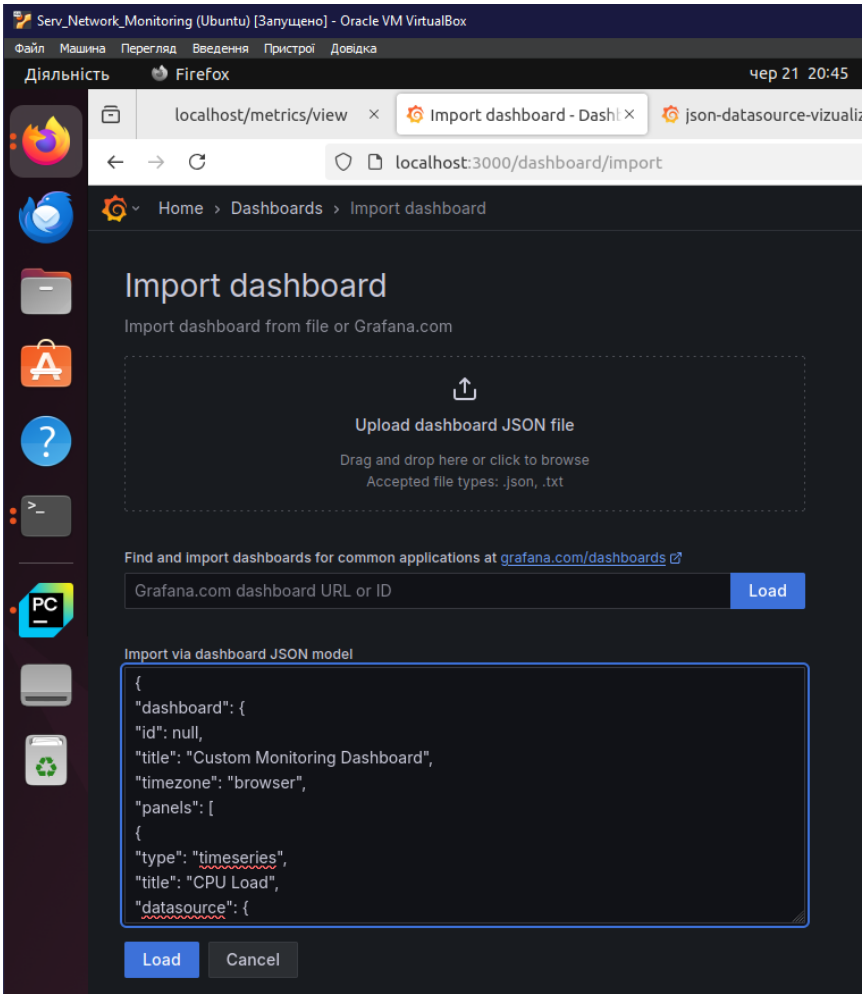


Рисунок 3.23 – Використання налаштованого JSON API

У цьому JSON-файлі також задаються трансформації (Transformations), наприклад, зміна типу полів — перетворення timestamp у тип "Time" для часової осі та CPU, Memory або Disk у числові значення ("Number"). Завдяки цьому при імпорті автоматично застосовуються всі необхідні параметри для коректного відображення метрик без потреби ручного налаштування кожного елементу dashboard:

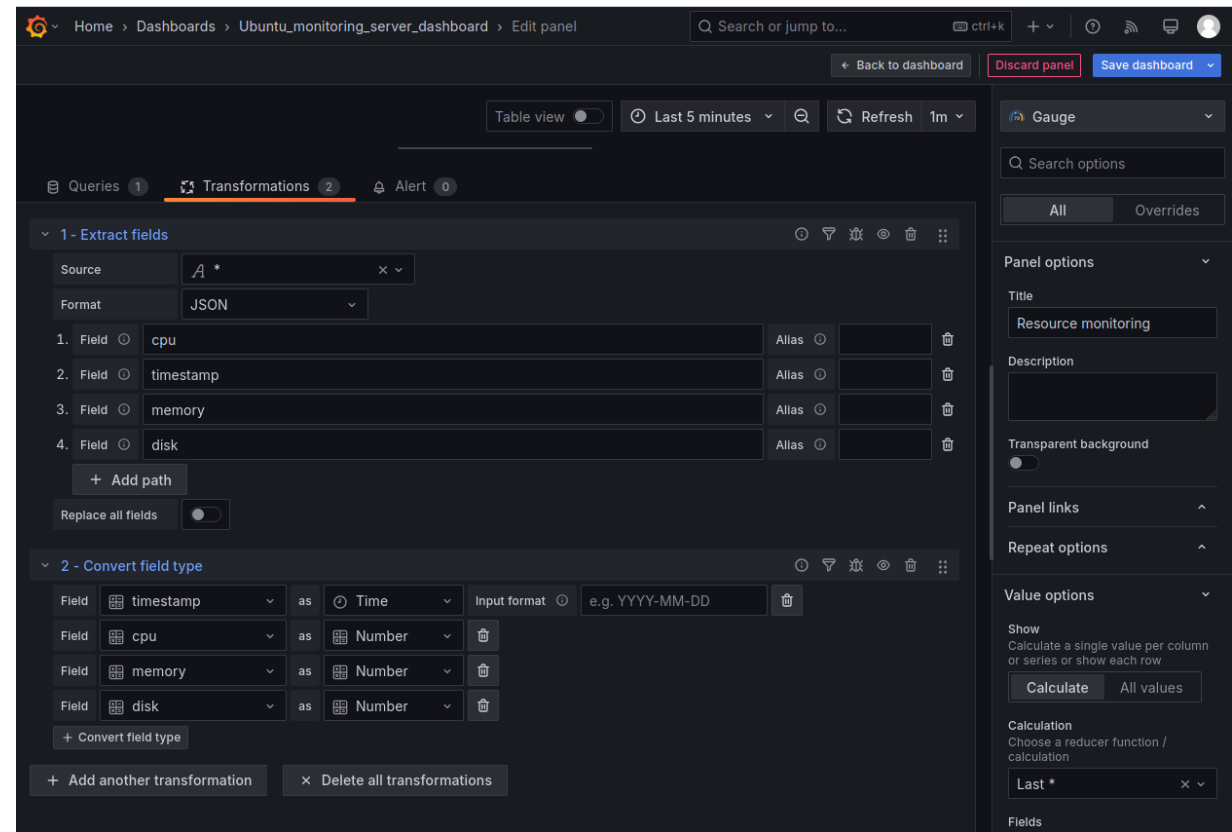


Рисунок 3.24 – Задання трансформацій для змінних

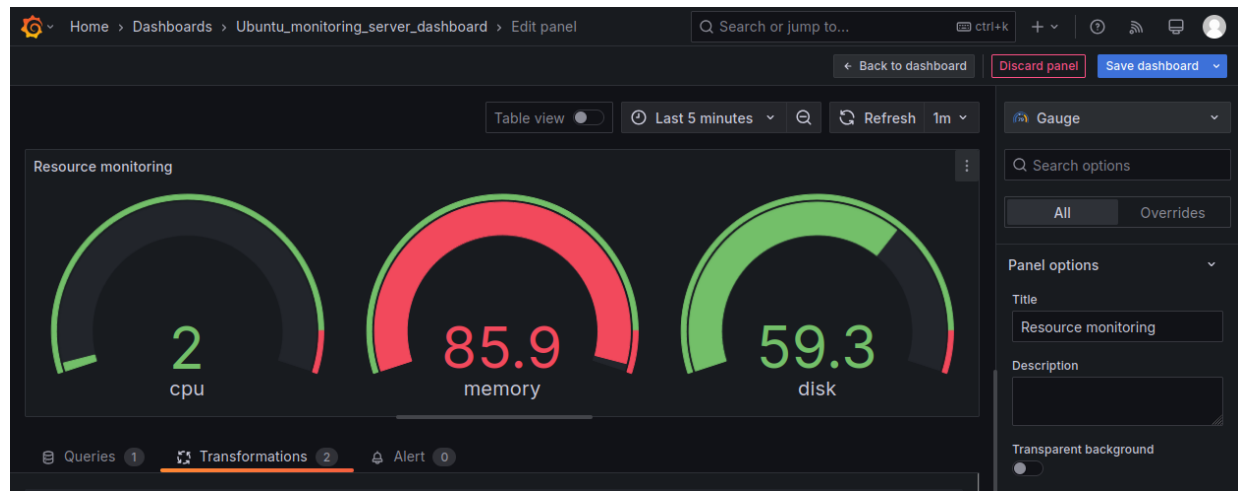


Рисунок 3.25 – Отримання метрик в реальному часі

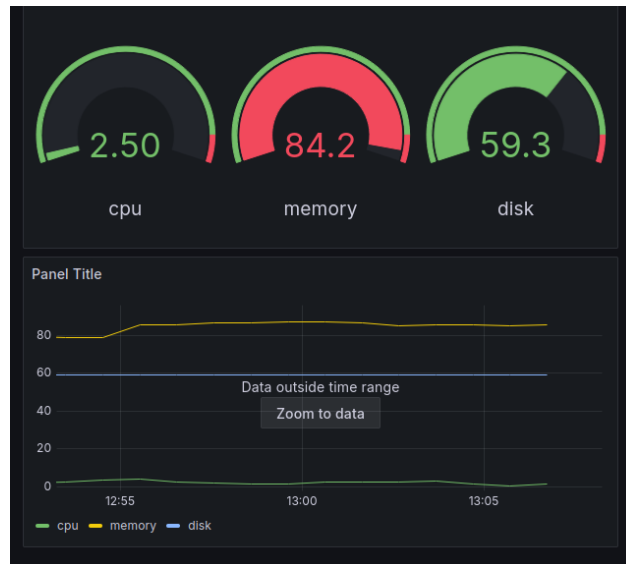


Рисунок 3.26 – Перегляд всіх метрик за останній час

Висновки до третього розділу

У результаті реалізації програмного агента моніторингу було розроблено ефективну систему збору, обробки та візуалізації метрик, яка дозволяє здійснювати контроль стану системних ресурсів хоста. Використання Nginx і Flask дозволяє забезпечити безпечну, модульну та масштабовану архітектуру програмного агента моніторингу з підтримкою веб-доступу до зібраних даних.

Завдяки використанню Python-агента забезпечено регулярне отримання інформації про навантаження на процесор, оперативну пам'ять та дисковий простір, що зберігається у базі даних SQLite для подальшого аналізу. Механізм виявлення аномальної активності реалізовано через перевірку зібраних значень на перевищення встановлених порогів, з можливістю надсилання сповіщень адміністратору через Telegram-бота при виявленні критичних ситуацій. Інструмент Grafana із підключеним джерелом даних JSON API забезпечує наочну візуалізацію метрик у вигляді окремих графіків для кожного параметра, що дозволяє своєчасно виявляти тенденції та потенційні проблеми у роботі системи. Таким чином, побудована система моніторингу є гнучкою, розширюваною та придатною для використання в умовах реальних експлуатаційних середовищ.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи були досягнуті всі основні завдання, визначені на етапі постановки проблеми.

Було сформульовано чіткі вимоги до програмного агента — мінімальне ресурсне навантаження на клієнтські хости, простота розгортання, збір ключових показників системи (CPU, RAM, Disk), виявлення аномальної активності та своєчасне інформування адміністратора через Telegram-бота. Також було передбачено інтеграцію з Grafana.

У розділі з проєктування інфраструктури було створено логічну модель комп'ютерної мережі підприємства у GNS3, що включає побудову IP-адресації, маршрутизацію, налаштування NAT для доступу в Інтернет, організацію домену для централізованої автентифікації та управління. Це забезпечило надійну основу для роботи програмного агента у реальній мережі.

У результаті аналізу мов програмування було обґрунтовано вибір Python як основної мови завдяки її простоті, підтримці роботи з мережею, наявності численних бібліотек (Flask, SQLite, requests, psutil) та можливості легкої інтеграції з Grafana та Telegram API. Для передачі даних між агентом і сервером було використано протокол HTTPS для забезпечення захищеного обміну.

На сервері реалізовано Flask-додаток із безпечним доступом через Nginx та самопідписаний SSL-сертифікат. Передбачено збереження отриманих даних та надання їх для запитів Grafana.

Розроблена система здатна автоматично виявляти аномальні стани (перевантаження CPU, брак пам'яті або місця на диску) та своєчасно інформувати адміністратора.

Власне рішення ідеально підходить для індивідуальних завдань моніторингу невеликої кількості хостів або у навчальних цілях, забезпечує гнучкість, простоту та повний контроль над логікою роботи агента, однак поступається Prometheus та Zabbix у масштабованості, автоматизації та готових функціях для великих інфраструктур.

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Огляд Zabbix моніторингу | Server Agent Windows – Youtube. [Електронний ресурс] : Режим доступу : https://www.youtube.com/watch?v=UCU16v5Zx7A&ab_channel=IT-SkillsЗаписьВІТкачалку (дата звернення: 08.06.2025)
2. 9 найкращих програмних інструментів для моніторингу мережі 2025, червень (безкоштовні та платні) – Promethues. [Електронний ресурс] : Режим доступу : <https://googiest.com/uk/блосі/найкраще-програмне-забезпечення-для-моніторингу-мережі/> (дата звернення: 08.06.2025)
3. Overview – Promethues. [Електронний ресурс] : Режим доступу : <https://prometheus.io/docs/introduction/overview/> (дата звернення: 09.06.2025)
4. Floor Plan Software – Planner5d. [Електронний ресурс] : Режим доступу: <https://planner5d.com/use/floor-plan-software> (дата звернення: 12.06.2025)
5. Що таке домен – hostpro. [Електронний ресурс] : Режим доступу: <https://hostpro.ua/wiki/ua/domains/what-is-domain/> (дата звернення: 14.06.2025)
6. Tanenbaum A. S., Wetherall D. J. Computer Networks. 5-те вид. 2011. 960 с. ISBN 978-0-13-212695-3.
7. Forouzan B. A. Data Communications and Networking. 5-те вид. New York: McGraw-Hill, 2012. 1264 с.
8. Sweigart A. Automate the Boring Stuff with Python. 2-ге вид. No Starch Press, 2019. 592 с. ISBN 978-1-59327-992-9.
9. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2-ге вид. O'Reilly Media, 2018. 258 с. ISBN 978-1-4919-3355-3.
10. When Should You Use HTTPS vs HTTP? – DashClicks Team. [Електронний ресурс] : Режим доступу: <https://www.dashclicks.com/blog/https-vs-http> (дата звернення: 15.06.2025)
11. Hunt C. TCP/IP Network Administration. 3-тє вид. O'Reilly Media, 2002. 748 с. ISBN 978-0-596-00334-0

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

- 12.Telegram Bot API. [Електронний ресурс] :Режим доступу:
<https://core.telegram.org/bots/api> (дата звернення: 16.06.2025)
- 13.Flask’s documentation. [Електронний ресурс] :Режим доступу:
<https://flask.palletsprojects.com/en/stable/> (дата звернення: 12.06.2025)
- 14.Owens M. The Definitive Guide to SQLite. 2-ге вид. Apress, 2010. 368 с.
- 15.nginx documentation. [Електронний ресурс] :Режим доступу:
<https://nginx.org/en/docs/> (дата звернення: 12.06.2025)
- 16.Gunicorn documentation. [Електронний ресурс] :Режим доступу:
<https://docs.gunicorn.org/en/stable/> (дата звернення: 12.06.2025)
- 17.GitLab documentation. [Електронний ресурс] :Режим доступу:
<https://docs.gitlab.com/> (дата звернення: 12.06.2025)
- 18.Swagger Editor. [Електронний ресурс] :Режим доступу:
<https://editor.swagger.io/> (дата звернення: 18.06.2025)
- 19.Webber J. REST in Practice: Hypermedia and Systems Architecture. Manning Publications, 2010. 400 с. ISBN 978-1-935182-63-0.
- 20.Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Doctoral dissertation. University of California, Irvine, 2000.
- 21.Grafana technical documentation. [Електронний ресурс] :Режим доступу: <https://grafana.com/docs/> (дата звернення: 12.06.2025)

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

Налаштування домену

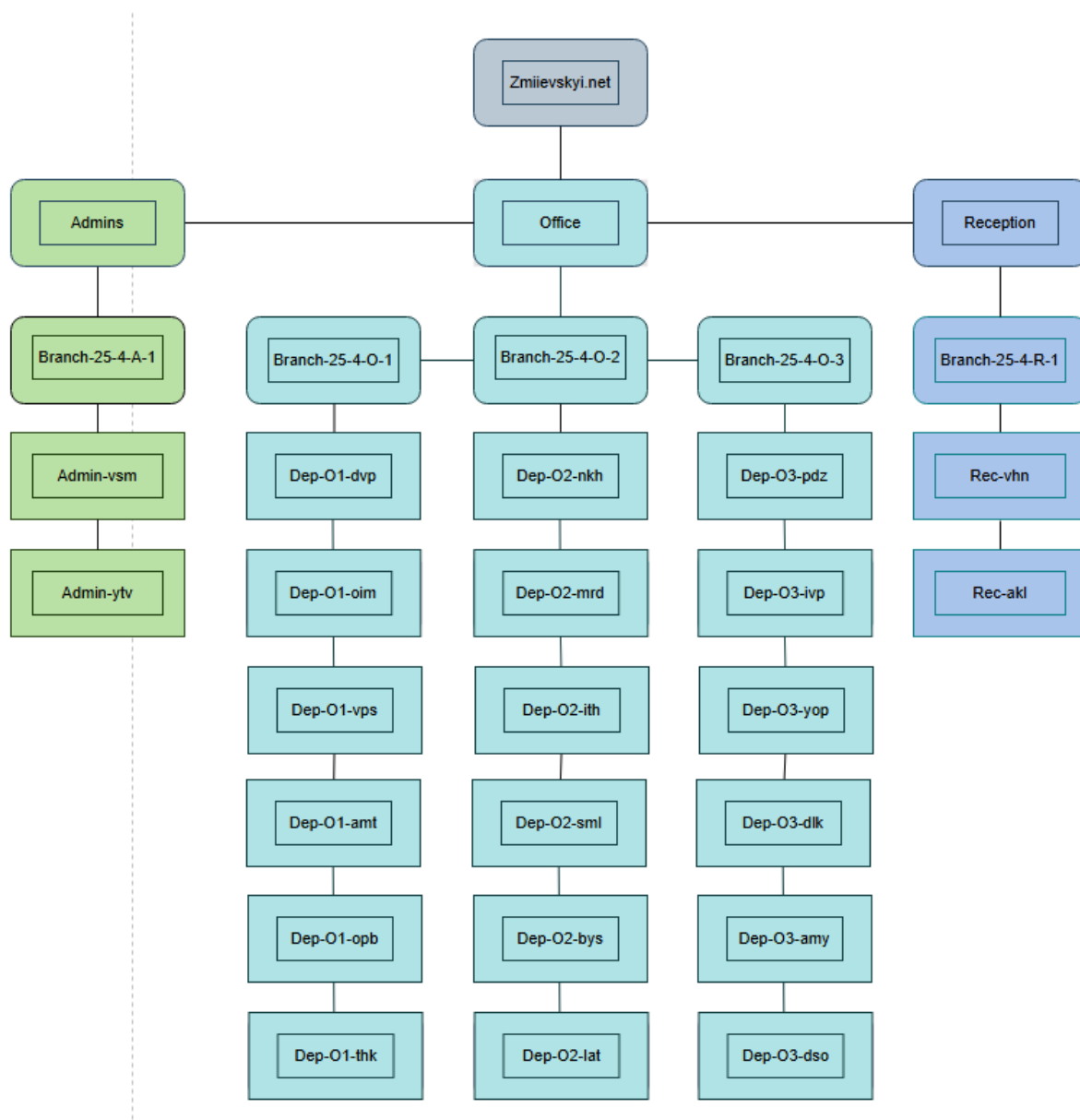


Рисунок А.1 – Схема OU користувачів

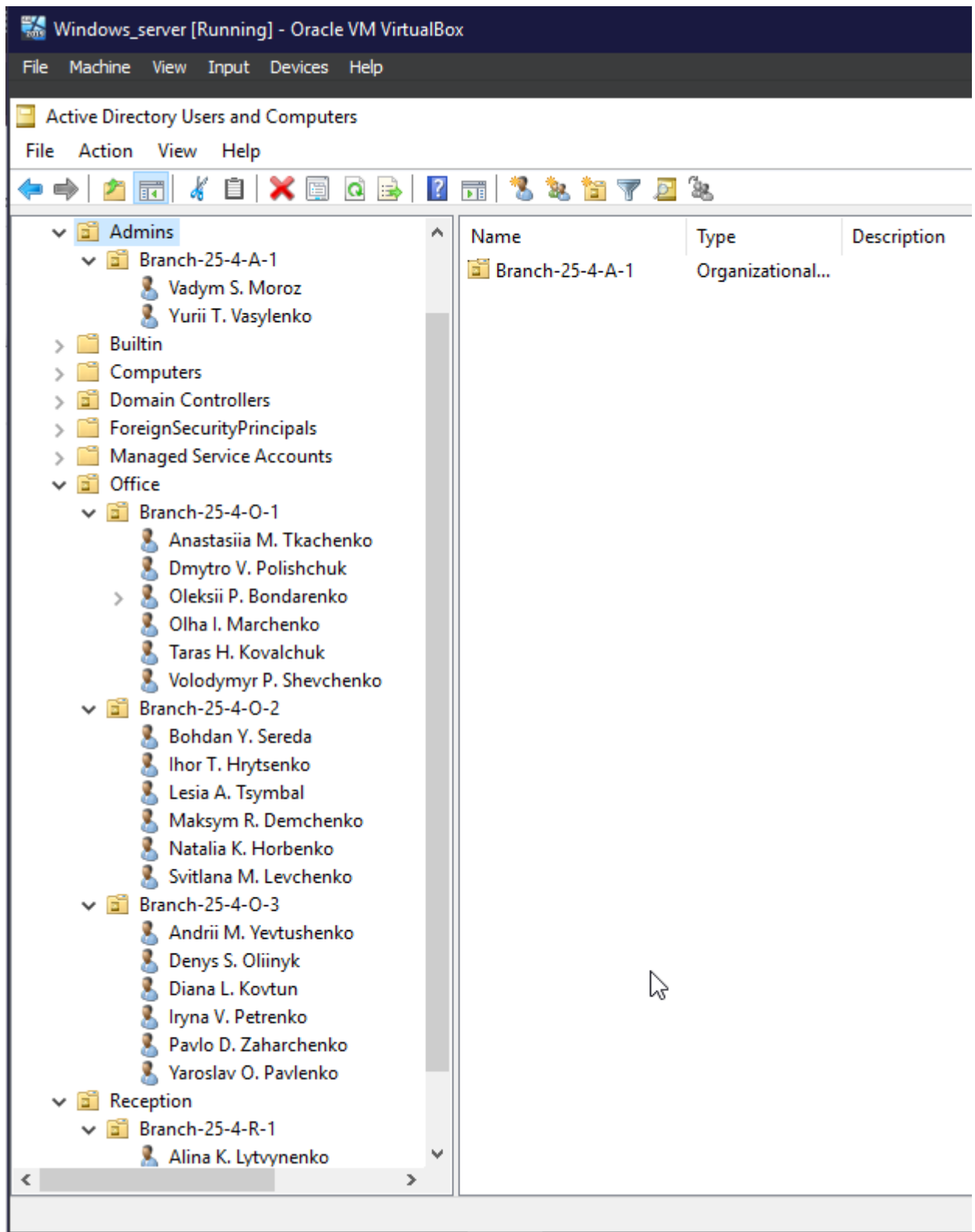


Рисунок А.2 – Створені користувачі на Windows Server

Топологія мережі

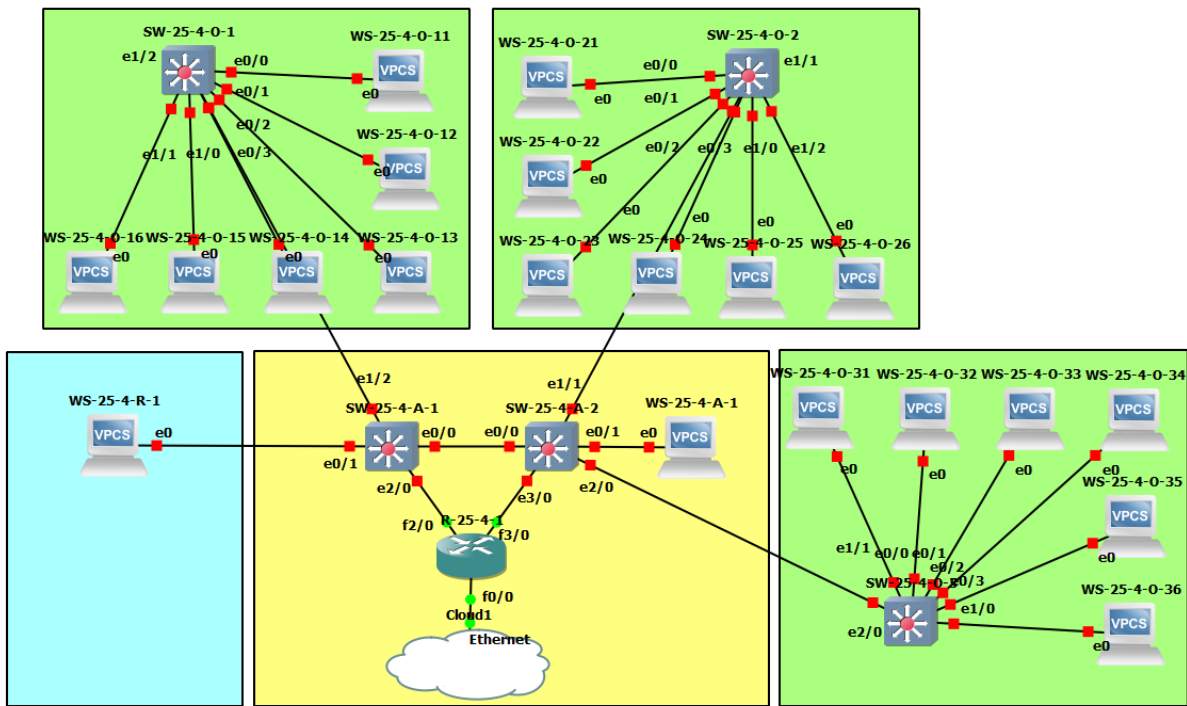


Рисунок Б.1 – Схема мережі

OpenAPI 3.0-специфікація

openapi: 3.0.3

info:

title: Monitoring Server API

version: 1.0.0

description: API для збору та перегляду системних метрик

servers:

- url: http://localhost:5000

paths:

/:

get:

summary: Статус сервера

responses:

'200':

description: Сервер працює

content:

text/plain:

schema:

type: string

example: Monitoring server is running.

/metrics:

get:

summary: Отримати інструкцію по використанню

responses:

'200':

description: Інструкція для POST-запиту

content:

application/json:

schema:

type: object

properties:

usage:

type: string

example: POST JSON with metrics

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

post:
  summary: Відправити метрики
  requestBody:
    required: true
    content:
      application/json:
        schema:
          type: object
          properties:
            timestamp:
              type: integer
            hostname:
              type: string
            os:
              type: string
            cpu:
              type: number
            memory_total:
              type: integer
            memory_used:
              type: integer
            memory_percent:
              type: number
            disk_total:
              type: integer
            disk_used:
              type: integer
            disk_percent:
              type: number
          required:
            - timestamp
            - hostname
            - cpu
            - memory_total
            - memory_used
            - memory_percent
            - disk_total
            - disk_used
            - disk_percent
  responses:
    '200':
      description: Дані збережено
      content:
        application/json:

```

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

schema:

type: object
properties:
 status:
 type: string
 example: stored
 alerts:
 type: array
 items:
 type: string

'400':

description: Відсутній JSON

content:

application/json:

schema:

type: object
properties:
 error:
 type: string
 example: No JSON

/metrics/view:

get:

summary: Перегляд останніх метрик

responses:

'200':

description: Повертає список останніх метрик

content:

application/json:

schema:

type: array
items:
 type: object
 properties:
 timestamp:
 type: integer
 hostname:
 type: string
 cpu:
 type: number
 mem:
 type: number
 disk:
 type: number

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

/metrics/grafana:

get:

summary: Отримати метрики для Grafana

responses:

'200':

description: Повертає останні 100 записів у форматі для Grafana

content:

application/json:

schema:

type: array

items:

type: object

properties:

timestamp:

type: integer

description: У мілісекундах

hostname:

type: string

cpu:

type: number

memory:

type: number

disk:

type: number

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

Інтеграція бази даних SQLite

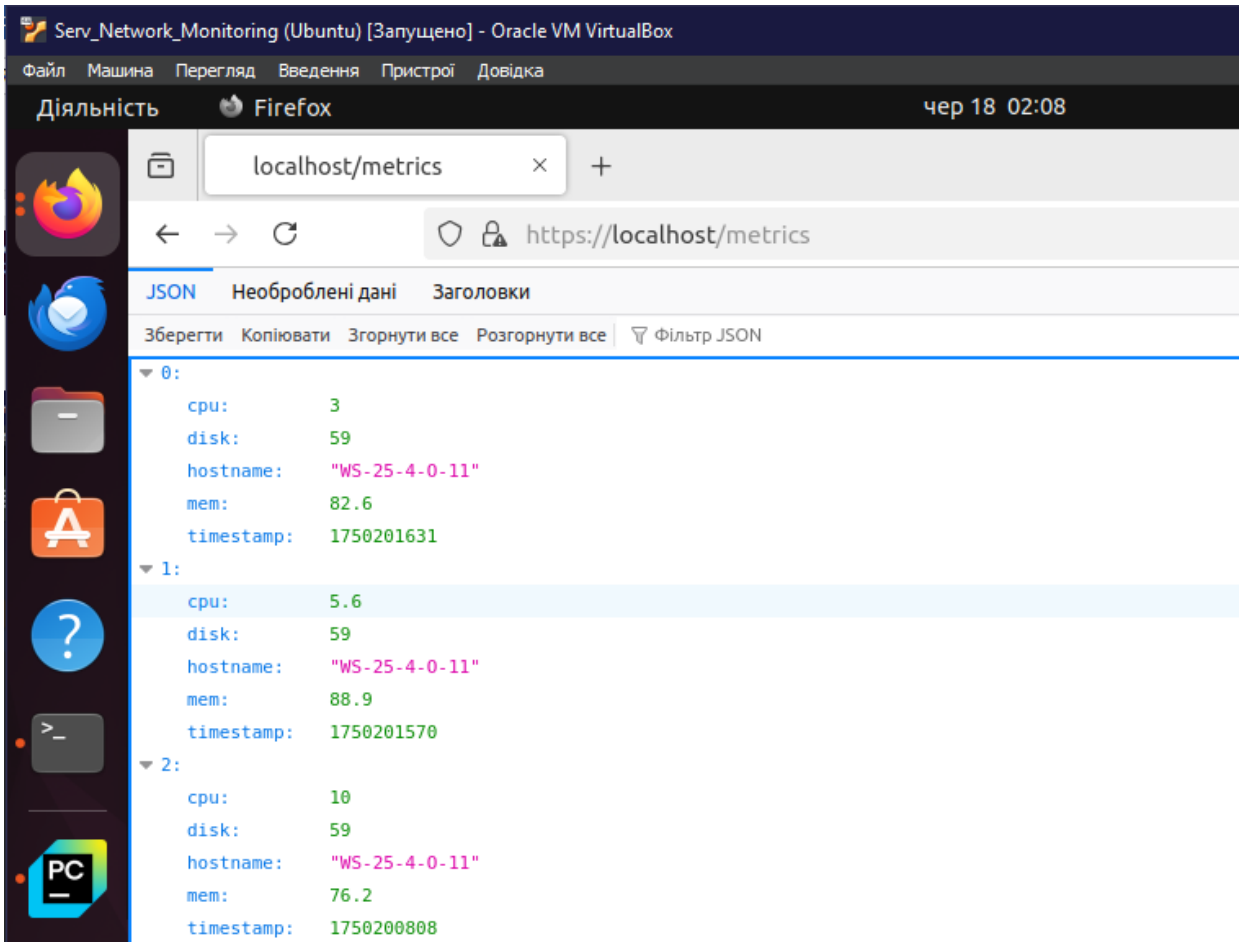


Рисунок Г.1 – Перегляд отриманих метрик з бази даних

Виявлення аномальної активності

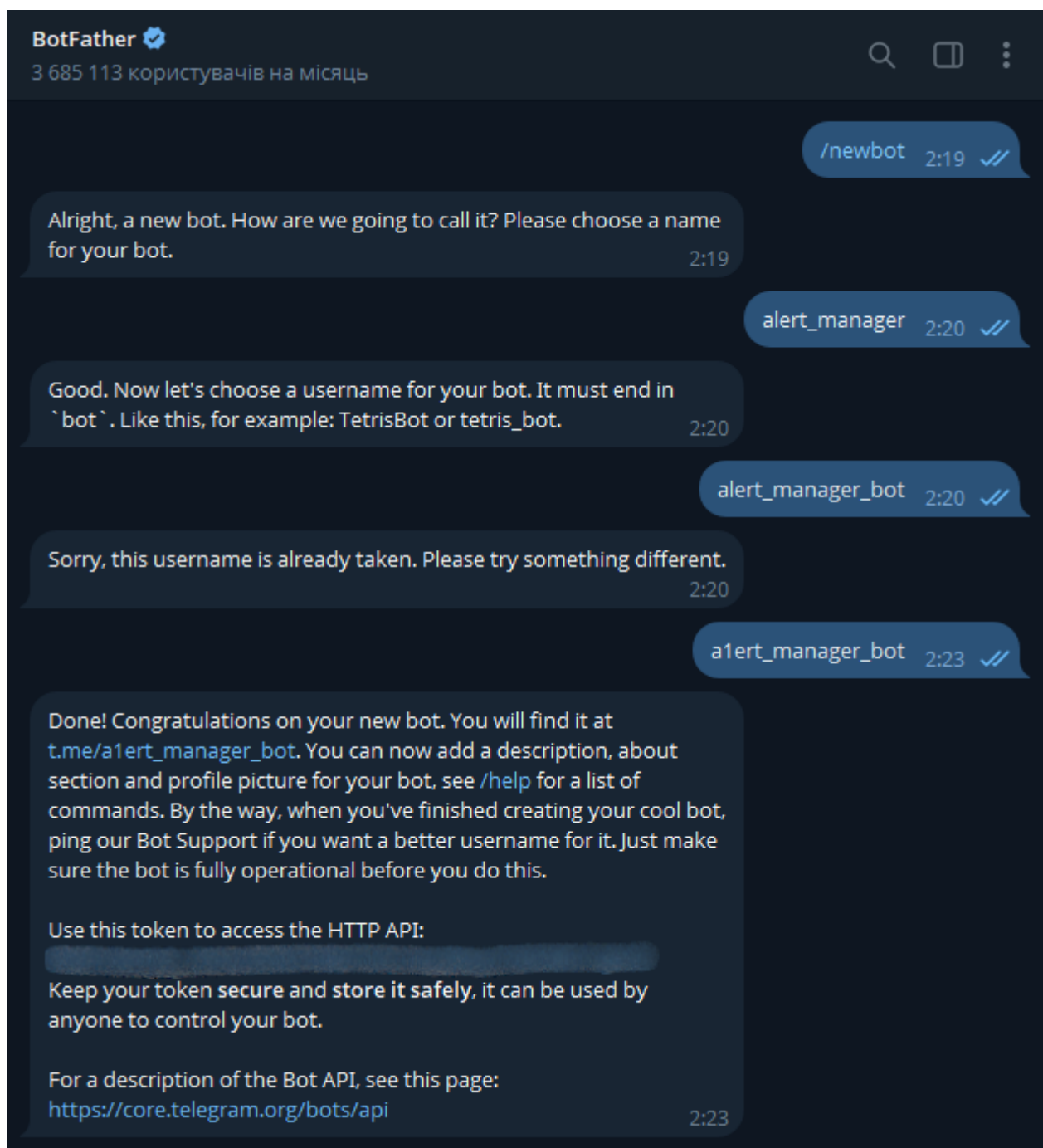


Рисунок Д.1 – Створення боту, отримання токену

		Змієвський В.В.			ККІКБ.КР.Б – 125 – 4 – ПЗ	Арк.
		Єфіменко А.А.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

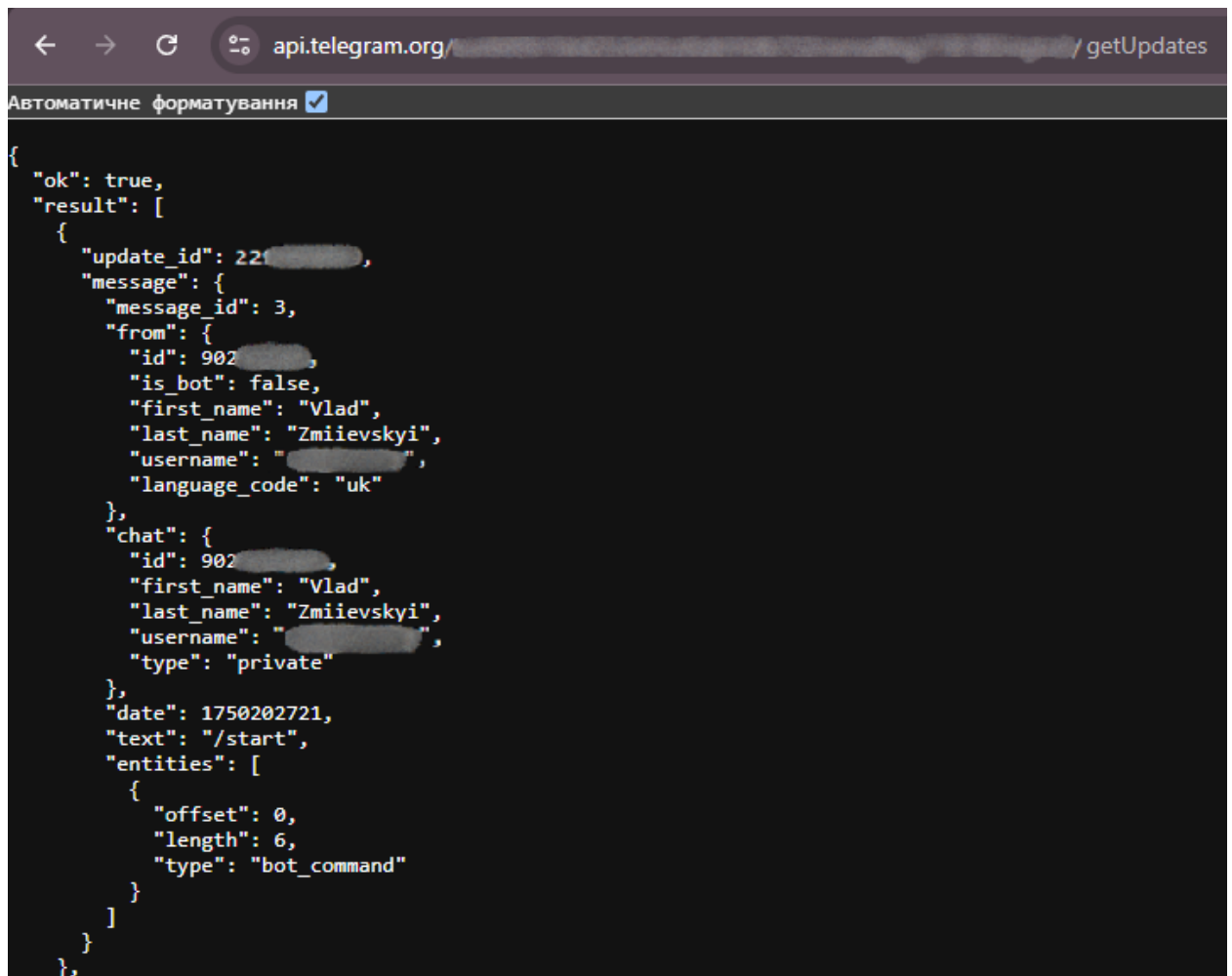


Рисунок Д.2 – Отримання chat id

Реалізація механізму авторизації для захищеного доступу до сторінок моніторингового сервера

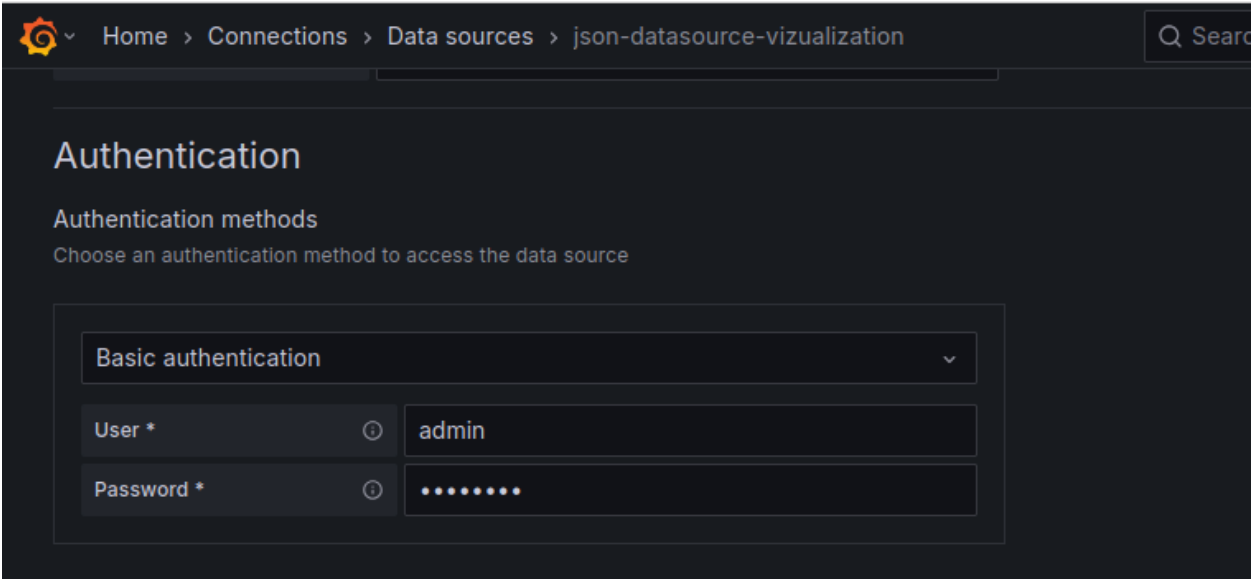


Рисунок Е.1 – Налаштування авторизації в Grafana

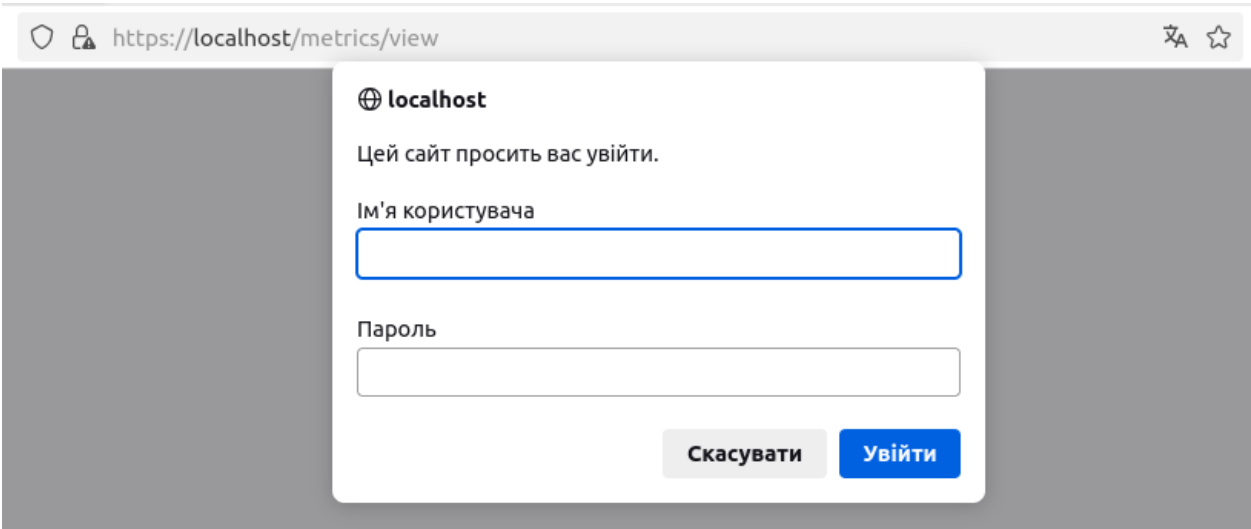


Рисунок Е.2 – Ауторизація для перегляду метрик на сервері