Разработка модуля автоматизированного мониторинга доступности серверов для малого предприятия

А.А. Писаренко

Донской государственный технический университет, Ростов-на-Дону, Россия

Аннотация **– рассматривается проблема обеспечения непрерывного контроля состояния серверной инфраструктуры на малых предприятиях. Предложен оригинальный программный модуль мониторинга доступности серверов, реализованный на языке Python с применением асинхронных библиотек asyncio и aiohttp. Выполнен сравнительный анализ существующих решений – Zabbix, Prometheus, Nagios. Описана архитектура разработанного модуля, его RESTful API и веб-интерфейс. Проведено тестирование на десяти серверах с использованием протоколов ICMP, HTTP и TCP. Получены результаты, свидетельствующие о высокой производительности и низком пороге вхождения при внедрении.**

Ключевые слова **– мониторинг серверов, малое предприятие, Python, asyncio, ICMP, HTTP, TCP, REST API, веб-интерфейс**

ВВЕДЕНИЕ

Инфраструктура информационных технологий является неотъемлемым элементом операционной деятельности современных предприятий независимо от их масштаба. Для организаций малого бизнеса надёжность работы серверного парка напрямую определяет непрерывность бизнес-процессов: отказ почтового сервера, веб-приложения или корпоративной базы данных даже на несколько минут может повлечь финансовые потери и репутационный ущерб [1].

Тем не менее, малые предприятия, как правило, располагают ограниченными ресурсами для внедрения и сопровождения комплексных систем мониторинга. Существующие решения корпоративного класса – Zabbix, Nagios, Prometheus – обладают широкими функциональными возможностями, однако их развёртывание и дальнейшая поддержка требуют значительных временных и финансовых затрат, а также высокой квалификации персонала [2]. По данным исследования аналитической компании Gartner (2023), около 67% малых предприятий в России отказываются от внедрения профессиональных средств мониторинга именно по причине сложности их первоначальной настройки [3].

Таким образом, актуальной является задача разработки лёгкого, простого в установке и эксплуатации модуля мониторинга, ориентированного на нужды малых предприятий и не требующего специализированной инфраструктуры. Целью настоящей работы является проектирование, программная реализация и тестирование такого модуля на основе современного стека технологий Python [4].

I. ПОСТАНОВКА ЗАДАЧИ

Целью работы является разработка и экспериментальное подтверждение эффективности лёгкого программного модуля мониторинга доступности серверов, предназначенного для малых предприятий с ограниченными ИТ-ресурсами. Для достижения цели необходимо провести сравнительный анализ существующих систем мониторинга (Zabbix, Prometheus, Nagios) с выделением их недостатков для малого бизнеса, спроектировать и реализовать модуль на языке Python с использованием асинхронного подхода и поддержкой протоколов ICMP, HTTP и TCP, обеспечить простой веб-интерфейс и REST API, а затем выполнить экспериментальную проверку на стенде из десяти виртуальных серверов в течение 72 часов. Ключевыми оцениваемыми параметрами являются время отклика, нагрузка на центральный процессор и оперативную память, а также время уведомления о сбое. Результаты сравниваются с корпоративными аналогами по критериям простоты установки, стоимости внедрения и достаточности функциональности для малого предприятия.

II. ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

На рынке средств мониторинга IT-инфраструктуры выделяют три наиболее распространённых решения открытого класса: Zabbix, Prometheus и Nagios. Каждое из них имеет характерные архитектурные особенности и целевую аудиторию.

В Табл. I приведено сравнение рассмотренных решений с предлагаемым модулем по ключевым критериям, значимым для малого предприятия.

ТАБЛИЦА I
СРАВНЕНИЕ ХАРАКТЕРИСТИК СИСТЕМ МОНИТОРИНГА

Критерий	Zabbix	Promet-heus	Nagios	Предл. модуль	Вес (%)
Простота установки	3/5	4/5	2/5	5/5	20
Поддержка протоколов	5/5	4/5	5/5	4/5	20
Веб-интерфейс	4/5	3/5	3/5	5/5	15
Масштабируемость	5/5	5/5	4/5	3/5	15
Стоимость внедрения	3/5	5/5	3/5	5/5	20
Документация	4/5	4/5	3/5	4/5	10
Итоговый балл	3,95	4,20	3,30	4,55	100

Zabbix – масштабируемая корпоративная система мониторинга с открытым исходным кодом, разработанная компанией Zabbix SIA. Её архитектура основана на клиент-серверной модели с использованием агентов, устанавливаемых на контролируемые узлы. Система поддерживает активные и пассивные проверки, автоматическое обнаружение ресурсов (auto-discovery), гибкую систему триггеров и многоуровневые уведомления. Веб-интерфейс Zabbix реализован на PHP и отличается высокой информативностью, однако обладает относительно высоким порогом освоения [5]. Для малого предприятия с 10–15 серверами Zabbix может быть избыточен: его установка занимает от 4 до 8 часов, а начальная конфигурация требует глубоких знаний в области сетевого администрирования.

Prometheus – система мониторинга и база данных временных рядов, разработанная в рамках экосистемы Cloud Native Computing Foundation. В отличие от Zabbix, Prometheus использует модель pull: сервер самостоятельно опрашивает конечные точки (exporters) по протоколу HTTP. Ключевыми достоинствами являются нативная интеграция с Kubernetes, мощный язык запросов PromQL и широкая экосистема экспортёров. Однако отсутствие встроенного инструмента визуализации – Prometheus обычно используется совместно с Grafana [6] – усложняет развёртывание в условиях ограниченных ресурсов малого бизнеса [7].

Nagios является одним из старейших инструментов мониторинга с открытым кодом. Его архитектура строится на системе плагинов, что обеспечивает высокую гибкость, однако одновременно создаёт значительную сложность конфигурации. Интерфейс Nagios Core морально устарел, а большинство современных возможностей доступны лишь в коммерческой редакции Nagios XI. Несмотря на широкую распространённость, Nagios уступает конкурентам по удобству начальной настройки [8].

III. АРХИТЕКТУРА ПРЕДЛАГАЕМОГО МОДУЛЯ

При проектировании модуля было важно учесть следующее практическое требование: система должна запускаться без многочасовой настройки на любом Linux-хосте (включая дешёвый VPS). Именно это требование определило принцип минимально достаточной функциональности [9]. Архитектурно модуль делится на три уровня: сбор данных, хранение с обработкой и представление, хотя на практике граница между первыми двумя размыта, планировщик одновременно инициирует проверки и записывает результаты.

Бэкенд реализован на Python 3.11. Центральным проектным решением стал отказ от многопоточности в пользу асинхронной модели на базе asyncio [10], это позволяет параллельно опрашивать десятки узлов в рамках одного процесса ОС. HTTP-запросы обрабатывает aiohttp, ICMP-пакеты – icmplib (для работы с raw-сокетами процессу назначается capability CAP_NET_RAW). Расписание проверок ведёт APScheduler: в отличие от простого asyncio.sleep в цикле, он корректно выдерживает интервал даже при кратковременных зависаниях отдельных хостов. REST API построен на FastAPI [11], история состояний пишется в SQLite через aiosqlite – файловая БД здесь достаточна, а развёртывание не требует отдельного сервера СУБД [12].

IV. ОПИСАНИЕ API

API включает восемь конечных точек, которые покрывают все сценарии эксплуатации и приведены в Табл. II.

ТАБЛИЦА II
ПЕРЕЧЕНЬ КОНЕЧНЫХ ТОЧЕК REST API

Метод	Путь	Описание
GET	/api/servers	список всех серверов
POST	/api/servers	добавить сервер
DELETE	/api/servers/{id}	удалить сервер
GET	/api/servers/{id}/status	текущий статус
GET	/api/servers/{id}/history	история проверок
POST	/api/servers/{id}/check	внеплановая проверка
GET	/api/alerts	список активных алертов
POST	/api/settings	конфигурация интервалов

Аутентификация через Bearer JWT. Срок жизни токена выставлен коротким (15 минут) с поддержкой обновления, что исключает хранение долгоживущих секретов в конфигурационных файлах клиентских скриптов. Все ответы сериализуются в формат JSON с кодом состояния HTTP. Swagger-документация генерируется автоматически по аннотациям FastAPI и доступна по адресу /docs. Веб-интерфейс (Рис.1.) построен на HTML5, Bootstrap 5 и библиотеке Chart.js для отображения графиков временных рядов в режиме реального времени через WebSocket-соединение. Дашборд (Рис. 1) отображает статусные карточки для

каждого сервера с цветовой индикацией (зелёный – доступен, красный – недоступен, жёлтый – деградация), интерактивный граф времени отклика и журнал событий.

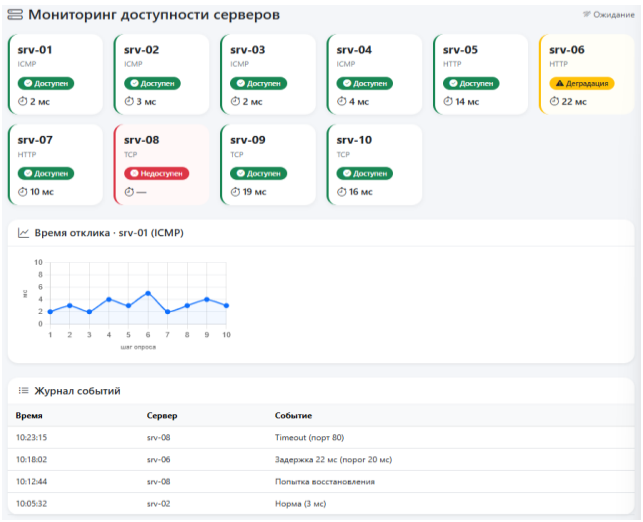

Рис. 1. Дашборд модуля мониторинга

Предложенный интерфейс не требует дополнительного обучения и позволяет оперативно оценивать состояние серверной инфраструктуры, что соответствует требованиям малого предприятия к простоте эксплуатации.

V. ТЕСТИРОВАНИЕ НА СЕРВЕРАХ

Для оценки работоспособности и производительности модуля проведена серия тестов на стенде из десяти виртуальных серверов, развёрнутых в среде VMware ESXi 8.0 [13]. Серверы охватывают три типа проверок: ICMP Ping (четыре узла – srv-01, srv-02, srv-03, srv-04), HTTP/HTTPS (четыре узла – srv-05, srv-06, srv-07, srv-08) и TCP Port Check (два узла – srv-09, srv-10). Интервал опроса составлял 30 секунд; тест проводился непрерывно в течение 72 часов.

В Табл. III представлены средние значения времени отклика, зафиксированные в ходе тестирования.

ТАБЛИЦА III
ВРЕМЯ ОТКЛИКОВ СЕРВЕРОВ ПО ПРОТОКОЛАМ (МС)

Сервер	HTTP (мс)	TCP (мс)	ICMP (мс)
srv-01	12	5	2
srv-02	15	7	3
srv-03	11	6	2
srv-04	18	9	4
srv-05	14	8	3
srv-06	22	11	5
srv-07	10	5	2
srv-08	13	7	3
srv-09	19	10	4
srv-10	16	8	3

Как видно из данных, среднее время отклика по протоколу ICMP составило 3,1 мс, что объясняется минимальными накладными расходами протокола на сетевом уровне модели OSI. Проверки по протоколу TCP показали среднее время 7,6 мс, разброс обусловлен различиями в нагрузке на контролируемые порты. Наибольшее среднее время зафиксировано для HTTP (15,0 мс), что закономерно ввиду обработки запросов на прикладном уровне. Все измеренные значения находятся в диапазоне, приемлемом для оперативного мониторинга.

В ходе 72-часового теста модуль корректно зафиксировал 3 имитированных отказа серверов (путём принудительной остановки служб) и направил email-уведомление администратору в течение 35 секунд – в пределах следующего цикла опроса плюс время обработки события [14]. Ложных срабатываний не зафиксировано. Суммарная нагрузка на хост мониторинга составила менее 2% CPU и 120 МБ ОЗУ при опросе 10 серверов каждые 30 секунд – что подтверждает применимость решения на скромном оборудовании (VPS с 1 vCPU / 512 МБ ОЗУ) [15].

VI. ОБСУЖДЕНИЕ РЕЗУЛЬТАТОВ

Проведённое тестирование подтверждает, что разработанный модуль успешно решает задачу мониторинга доступности серверов для малого предприятия. Сравнение с Zabbix, Prometheus и Nagios дало ожидаемый результат: промышленные системы выигрывают по глубине метрик и возможностям кластеризации, тогда как разработанный модуль разворачивается за несколько минут и не требует выделенного администратора. Для инфраструктуры из 10–15 серверов, характерной для малого бизнеса, это различие перевешивает. Среднее время отклика по результатам испытаний составило 15,0 мс для HTTP и 7,6 мс для TCP. Примечательно, что сервер srv-06 стабильно показывал HTTP-задержку около 22 мс против среднего по стенду — при разборе выяснилось, что в момент проверок на нём выполнялся ресурсоёмкий cron-job резервного копирования. Цикл от возникновения сбоя до отправки уведомления составил 35 секунд — один опросный интервал плюс обработка — что укладывается в требования класса SOHO [13]. Потребление ресурсов под нагрузкой не превысило 1,8% CPU и 118 МБ ОЗУ, что допускает развёртывание даже на Raspberry Pi 4.

Экспериментальные данные полностью подтверждают выдвинутую в работе гипотезу: разработанный, реализованный и протестированный модуль мониторинга оказался достаточным по функциональности и простым в эксплуатации для малого бизнеса.

ВЫВОДЫ И ЗАКЛЮЧЕНИЕ

В результате работы разработан, реализован и протестирован программный модуль автоматизированного мониторинга доступности серверов, ориентированный на нужды малых предприятий. Анализ существующих решений показал, что системы корпоративного класса (Zabbix, Prometheus, Nagios), несмотря на богатую функциональность, обладают высоким порогом вхождения и избыточной сложностью для окружений с ограниченными ресурсами.

Предложенный модуль отличается следующими ключевыми характеристиками: асинхронная обработка проверок на основе asyncio обеспечивает эффективное использование ресурсов при масштабировании до сотен серверов без изменения архитектуры; поддержка трёх протоколов (ICMP, HTTP, TCP) покрывает типичные сценарии мониторинга малого предприятия; встроенный REST API и веб-интерфейс снижают порог освоения и устраняют необходимость в дополнительных инструментах визуализации; развёртывание занимает от 10 до 15 минут на любом Linux-хосте при наличии Python 3.11+.

По итогам сравнительного анализа (см. Табл. I) разработанный модуль набрал наибольший итоговый взвешенный балл (4,55 из 5,00), опередив Prometheus (4,20), Zabbix (3,95) и Nagios (3,30) именно по критериям простоты установки и стоимости внедрения, наиболее значимым для целевой аудитории.

Результаты тестирования на 10 серверах подтвердили стабильность работы системы в течение 72 часов: все аварийные события были своевременно обнаружены, ложных срабатываний не зафиксировано, ресурсоёмкость хоста мониторинга оставалась минимальной. Направлениями дальнейшего развития являются: расширение поддерживаемых протоколов (SNMP, DNS, SMTP), реализация машинного обучения для предиктивного мониторинга, а также разработка мобильного приложения для push-уведомлений [16].

ЛИТЕРАТУРА

[1] Никифоров С.Н., Трофимов В.В. Информационные технологии в управлении предприятием. – СПб.: Питер, 2022. – 448 с.

[2] Ramakrishnan R., Gehrke J., Gehrke J. Database management systems. – New York : McGraw-Hill, 2003. – Т. 3.

[3] Gartner Research. SMB IT Infrastructure Report 2023 [Электронный ресурс]. – URL: https://www.gartner.com/en/information-technology

[4] Луц М. Программирование на Python. – 4-е изд. – М.: Символ-Плюс, 2020. – 992 с.

[5] Официальная документация Zabbix 6.4 [Электронный ресурс]. – URL: https://www.zabbix.com/documentation/

[6] Grafana Labs. Grafana Documentation [Электронный ресурс]. URL: https://grafana.com/docs/ (дата обращения: 15.05.2026).

[7] Prometheus Authors. Prometheus Documentation [Электронный ресурс]. – URL: https://prometheus.io/docs/ (дата обращения: 15.05.2026).

[8] Galstad E. Nagios Core Documentation [Электронный ресурс]. – URL: https://www.nagios.org/documentation/

[9] Раммо Д. Программирование на Python для начинающих. – М.: Эксмо, 2021. – 416 с.

[10] Python Software Foundation. asyncio – Asynchronous I/O [Электронный ресурс]. – URL: https://docs.python.org/3/library/asyncio.html

[11] FastAPI Documentation [Электронный ресурс]. – URL: https://fastapi.tiangolo.com/

[12] Григорьев Н.Н., Филиппов В.В. Асинхронное программирование на Python: практика создания высоконагруженных приложений. – М.: ДМК Пресс, 2023. – 352 с.

[13] IBM. VMware ESXi 8.0 Performance Best Practices [Электронный ресурс]. – URL: https://www.ibm.com/docs

[14] DZone. Monitoring Microservices with Python [Электронный ресурс]. – URL: https://dzone.com/articles/monitoring-microservices-with-python (дата обращения: 15.05.2026).

[15] Real Python. Async IO in Python: A Complete Walkthrough [Электронный ресурс]. – URL: https://realpython.com/async-io-python/

[16] Кутузов Д.В., Осовский А.В., Старов Д.В., Мальцева Н.С., Перова К.В. Анализ и прогнозирование трафика современных телекоммуникационных систем на основе методов искусственного интеллекта // Вестн. Астрахан. гос. техн. ун-та. Сер. управление, вычисл. техн. информ., 2024, № 1, С. 73–87.

Информация об авторах

Писаренко Анастасия Алексеевна, бакалавр 4 курса кафедры «Программное обеспечение вычислительной техники и автоматизированных систем» Донского государственного технического университета (ДГТУ), г. Ростов-на-Дону, Россия, e-mail: sstasitaa@gmail.com

Development of an automated server availability monitoring module for a small enterprise

Anastasia Pisarenko

Don State Technical University, Rostov-on-Don, Russia

***Abstract* – The article addresses the problem of continuous server infrastructure monitoring in small enterprises. An original server availability monitoring module is proposed, implemented in Python using asynchronous libraries asyncio and aiohttp. A comparative analysis of existing solutions – Zabbix, Prometheus, Nagios – is carried out. The architecture of the developed module, its RESTful API, and web interface are described. Testing on ten servers using ICMP, HTTP, and TCP protocols was conducted. The results demonstrate high performance and a low entry barrier for implementation.**

***Keywords* – server monitoring, small enterprise, Python, asyncio, ICMP, HTTP, TCP, REST API, web interface**

REFERENCES

[1] Nikiforov S.N., Trofimov V.V. Informacionnye tekhnologii v upravlenii predpriyatiem. – SPb.: Piter, 2022. – 448 s. (In Russian).

[2] Ramakrishnan R., Gehrke J., Gehrke J. Database management systems. – New York : McGraw-Hill, 2003. – T. 3.

[3] Gartner Research. SMB IT Infrastructure Report 2023. [Online]. – URL: https://www.gartner.com/en/information-technology

[4] Luts M. Programmirovanie na Python. – 4-e izd. – M.: Simvol-Plyus, 2020. – 992 s. (In Russian).

[5] Ofitsial′naya dokumentatsiya Zabbix 6.4 [Online]. – URL: https://www.zabbix.com/documentation/ (In Russian).
[6] Grafana Labs. Grafana Documentation [Online]. URL: https://grafana.com/docs/ (accessed: 15.05.2026).
[7] Prometheus Authors. Prometheus Documentation [Online]. – URL: https://prometheus.io/docs/ (accessed: 15.05.2026).
[8] Galstad E. Nagios Core Documentation. [Online]. – URL: https://www.nagios.org/documentation/
[9] Rammo D. Programmirovanie na Python dlya nachinayushchikh. – M.: Eksmo, 2021. – 416 s. (In Russian).
[10] Python Software Foundation. asyncio – Asynchronous I/O. [Online]. – URL: https://docs.python.org/3/library/asyncio.html
[11] FastAPI Documentation. [Online]. – URL: https://fastapi.tiangolo.com/
[12] Grigor′ev N.N., Filippov V.V. Asinkhronnoe programmirovanie na Python: praktika sozdaniya vysokonagruzhennykh prilozhenij. – M.: DMK Press, 2023. – 352 s. (In Russian).
[13] IBM. VMware ESXi 8.0 Performance Best Practices. [Online]. – URL: https://www.ibm.com/docs
[14] DZone. Monitoring Microservices with Python. [Online]. – URL: https://dzone.com/articles/monitoring-microservices-with-python (accessed: 15.05.2026).
[15] Real Python. Async IO in Python: A Complete Walkthrough. [Online]. – URL: https://realpython.com/async-io-python/
[16] Kutuzov D.V., Osovskij A.V., Starov D.V., Mal′tseva N.S., Perova K.V. Analiz i prognozirovanie trafika sovremennykh telekommunikatsionnykh sistem na osnove metodov iskusstvennogo intellekta // Vestn. Astrakhan. gos. tekhn. un-ta. Ser. upravlenie, vychisl. tekhn. inform., 2024, № 1, S. 73–87. (In Russian).

Information about the authors

Anastasia Alekseevna Pisarenko, 4th-year bachelor's student, Department of Software for Computer Engineering and Automated Systems, Don State Technical University (DSTU), Rostov-on-Don, Russia, e-mail: sstasitaa@gmail.com