NAPICONFIG

Веб-интерфейс управления Linux-системами для IoT и промышленных устройств

Эксплуатация экземпляра программного обеспечения

Документ 2

2026

СОДЕРЖАНИЕ

1. Назначение документа

Настоящий документ содержит информацию, необходимую для эксплуатации экземпляра программного обеспечения NAPICONFIG, предоставленного для проведения экспертной проверки. Документ основан на исходных кодах проекта и визуальных материалах.

Цели документа:

- описать базовые действия по запуску и проверке экземпляра ПО;
- перечислить ключевые компоненты для эксплуатации;
- указать основные места хранения конфигураций и логов;
- дать рекомендации для сопоставления рабочего экземпляра с приведенными изображениями экранов.

2. Область применения

Документ актуален для:

- экспертов, проводящих проверку работоспособности программного обеспечения;
- операторов, выполняющих эксплуатационные проверки установленного экземпляра;
- аудиторов, проводящих анализ соответствия заявленной архитектуры.

3. Структура развернутого экземпляра

Проект состоит из следующих основных частей:

Компонент	Назначение
napi-api	Backend, реализующий REST API на FastAPI
napi-ui	Frontend, Vue.js SPA для управления и мониторинга
system	Системные файлы для запуска сервисов, Nginx и сертификатов

Дополнительно используются каталоги `img/` и `img2/` со скриншотами интерфейса, служащими образцами для визуальной сверки.

4. Компоненты и основные файлы

4.1. Компонент napi-api

- `napi-api/app/main.py` — точка входа API;
- `napi-api/app/core/config.py` — конфигурация приложения;
- `napi-api/app/core/database.py` — инициализация базы данных;

- `napi-api/app/modules/` — модули бизнес-логики: `auth`, `network`, `system`, `sensors`, `snmp`, `modbus_gw`, `terminal`, `system_stats`;
- `napi-api/system/backend/napi-api.service` — unit-файл systemd для запуска сервиса;
- `napi-api/update.sh` — вспомогательный скрипт обновления.

4.2. Компонент napi-ui

- `napi-ui/src/main.ts` — точка входа фронтенда;
- `napi-ui/src/router/` — маршруты приложения;
- `napi-ui/src/stores/` — хранилища Pinia;
- `napi-ui/src/components/` — основные визуальные компоненты;
- `napi-ui/src/client/` — автогенерированный API-клиент на основе OpenAPI;
- `napi-ui/package.json` — зависимости и скрипты сборки;
- `napi-ui/update.sh` — вспомогательный скрипт обновления.

4.3. Компонент system

- `system/cert/` — SSL-сертификаты и ключи;
- `system/nginx/nginx.conf` — конфигурация обратного прокси;
- `system/backend/` — описания systemd-сервисов;
- `system/README.md` — общие указания по системной части.

5. Системные требования и подготовка

5.1. Требования

- операционная система: Linux (Debian/Ubuntu или совместимая);
- Python 3.10+ для backend;
- Node.js 18+ и npm для сборки frontend;
- systemd для управления сервисами;
- Nginx или другой веб-сервер для проксирования frontend (опционально).

5.2. Подготовка

- убедиться, что развернут полный пакет с каталогами `napi-api`, `napi-ui`, `system`;
- проверить настройки сети и доступность необходимых портов;
- проверить наличие прав администратора для управления сервисами.

6. Запуск экземпляра

6.1. Backend (napi-api)

Запуск в виде системного сервиса:

```
sudo cp /opt/napiconfig/napi-api/system/backend/napi-api.service
/etc/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable napi-api.service
sudo systemctl start napi-api.service
sudo systemctl status napi-api.service
```

Проверка доступности API:

```
curl -I http://localhost:8000/docs
```

6.2. Frontend (napi-ui)

Развертывание вручную:

```
cd /opt/napiconfig/napi-ui
npm install
npm run build
```

Запуск dev-сервера для проверки:

```
npm run dev -- --host 0.0.0.0 --port 5173
```

Стандартные адреса проверки: `http://localhost:5173` для dev-режима, `https://<host>/` для продакшен-сборки через Nginx.

6.3. Nginx и SSL

- `system/nginx/nginx.conf` содержит примеры проксирования к API и UI;
- `system/cert/` содержит сертификаты, используемые для HTTPS;
- Nginx запускается стандартными командами `systemctl enable nginx` и `systemctl start nginx`.

7. Проверка работоспособности

- API должен возвращать документацию OpenAPI по адресу `http://localhost:8000/docs`;
- frontend должен загружаться и показывать главное окно (панель мониторинга);
- реальный интерфейс следует сверять с изображениями экранов, приведенными в разделе 12 настоящего документа.

8. Основные пользовательские разделы интерфейса

8.1. Главная панель (Dashboard)

- состояние системы;

- статистика CPU, памяти и диска;
- текущие сетевые параметры.

8.2. Управление сервисами

- просмотр списка systemd-сервисов;
- запуск, остановка и перезапуск служб;
- фильтрация и просмотр статусов.

8.3. Сетевые настройки

- просмотр интерфейсов;
- настройка IP-адресов и маршрутов;
- задание DNS;
- управление VLAN и беспроводными соединениями.

8.4. Датчики и телеметрия

- настройка источников данных;
- тестирование конфигураций;
- просмотр статистики датчиков;
- сохранение конфигураций Telegraf/InfluxDB.

8.5. Веб-терминал

Обеспечивает удаленный доступ к командной оболочке через браузер по защищенному соединению с одноразовыми токенами доступа.

9. Диагностика и логирование

9.1. Логи systemd

```
sudo journalctl -u napi-api.service -n 200 --no-pager
sudo journalctl -u nginx -n 200 --no-pager
```

9.2. Системные логи

- `/var/log/syslog`;
- `/var/log/messages` (если доступен).

9.3. Логи приложения

- при работающем dev-сервере Vue.js логи выводятся в терминале;
- при запуске `napi-api` через systemd логи доступны через `journalctl`.

9.4. Проверка состояния сервисов

```
sudo systemctl status napi-api.service
sudo systemctl status nginx.service
sudo systemctl status docker.service   # если используется контейнеризация
```

10. Конфигурационные файлы для эксплуатации

Файл / каталог	Назначение
napi-api/app/core/config.py	Конфигурационные параметры backend
napi-api/system/backend/napi-api.service	Описание systemd-сервиса
napi-ui/package.json, napi-ui/vite.config.ts	Настройка фронтенда
system/nginx/nginx.conf	Прокси и SSL
system/cert/	Сертификаты

11. Рекомендации для экспертной проверки и контрольные точки

- проверку достаточно проводить на одном экземпляре `napi-api` и одном экземпляре `napi-ui`;
- экспертная оценка должна учитывать соответствие поведения интерфейса и функциональных разделов исходному коду;
- графический интерфейс следует сравнивать с изображениями раздела 12 для подтверждения визуального соответствия;
- не изменяйте состояние системы без фиксации: используйте отдельную копию проекта или снимайте снимки экрана для отчета.

Контрольные точки эксплуатации:

- backend запущен и отвечает на `/docs`;
- frontend доступен по тестовому URL;
- экраны соответствуют навигационной структуре и изображениям раздела 12;
- systemd-сервис `napi-api` активен и работает без ошибок;
- веб-терминал открывается, авторизация проходит успешно.

12. Описание экранов интерфейса

12.1. Главная страница (панель мониторинга)

Главная страница NAPICONFIG предоставляет обзор состояния системы: сводку по процессору, ОЗУ, свопу, температуре и дисковой подсистеме, статус сетевых интерфейсов, а также сведения об ОС, времени работы и текущей дате.

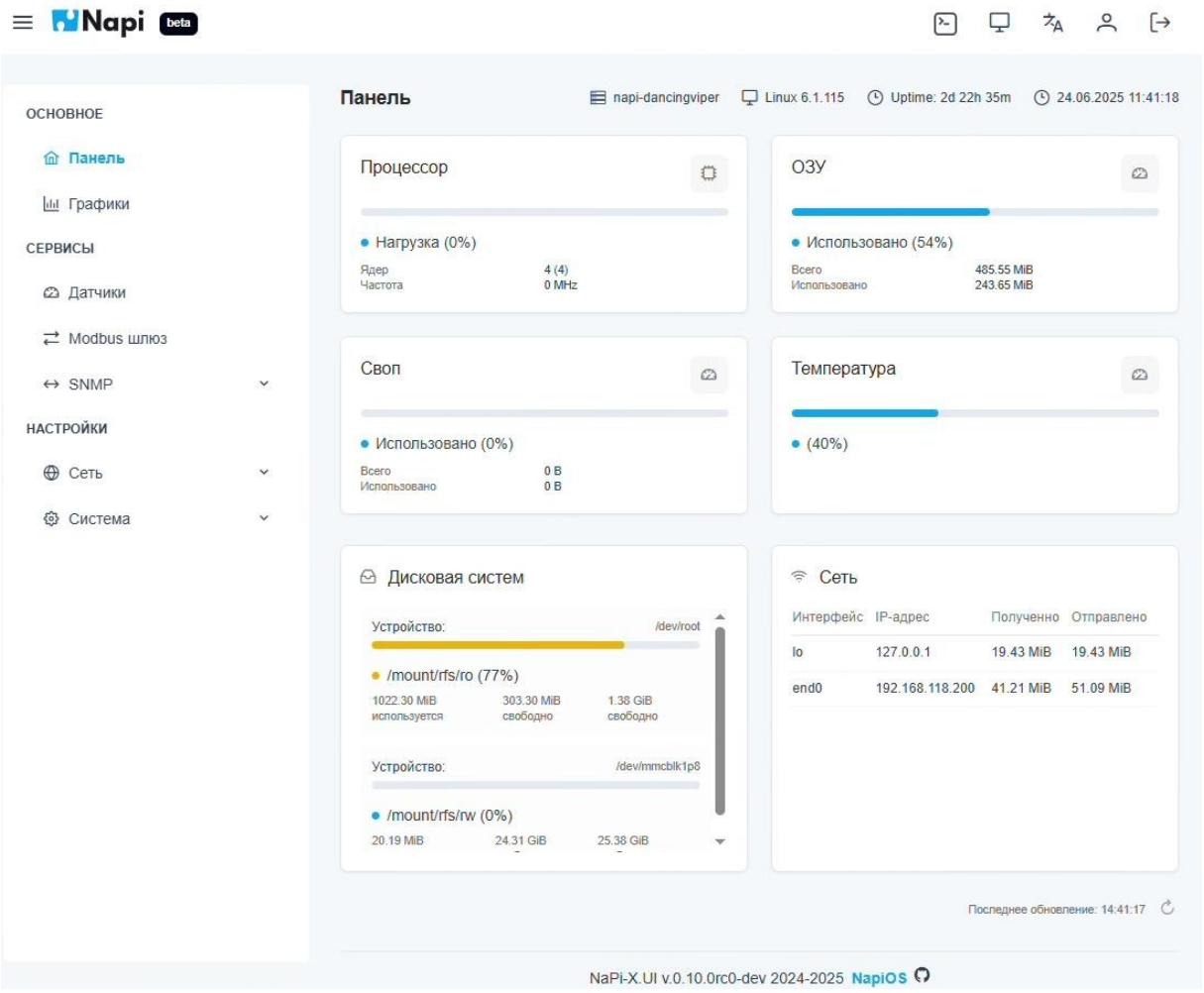

Рисунок 1 — Главная панель мониторинга

12.2. Управление сетевыми интерфейсами

Экран управления сетевыми интерфейсами позволяет просматривать статистику интерфейса (MAC-адрес, MTU, скорость, флаги, IPv4/IPv6-адреса, текущую скорость обмена), а также редактировать настройки: DHCP, IP-адреса, маршруты, DNS-серверы и MTU.

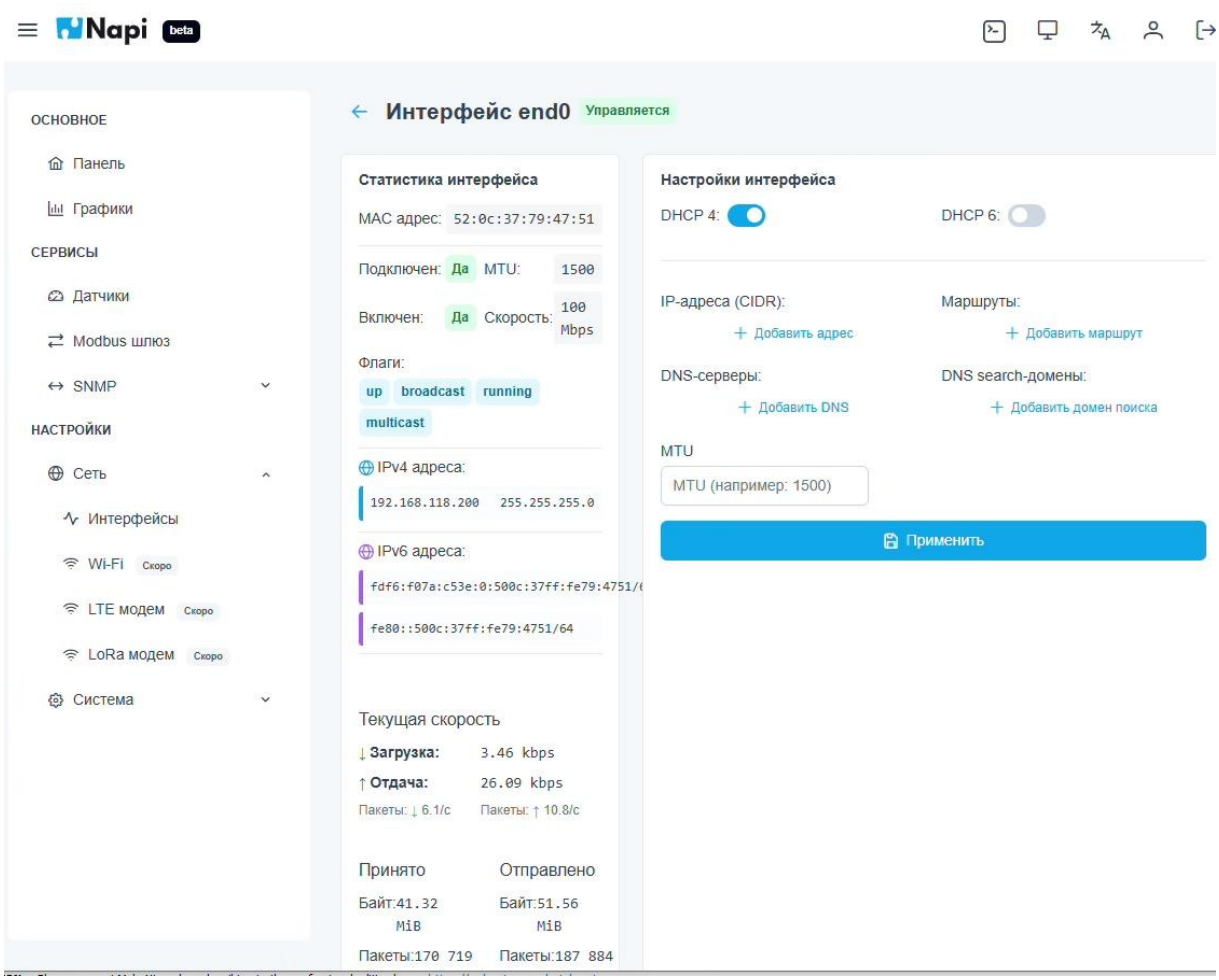

Рисунок 2 — Настройки сетевого интерфейса end0

12.3. Список датчиков

Раздел «Датчики» отображает зарегистрированные конфигурации датчиков, их статусы, производителей и метки. Здесь же доступно управление службой датчиков: остановка, перезапуск, активация, перезагрузка и очистка базы данных.

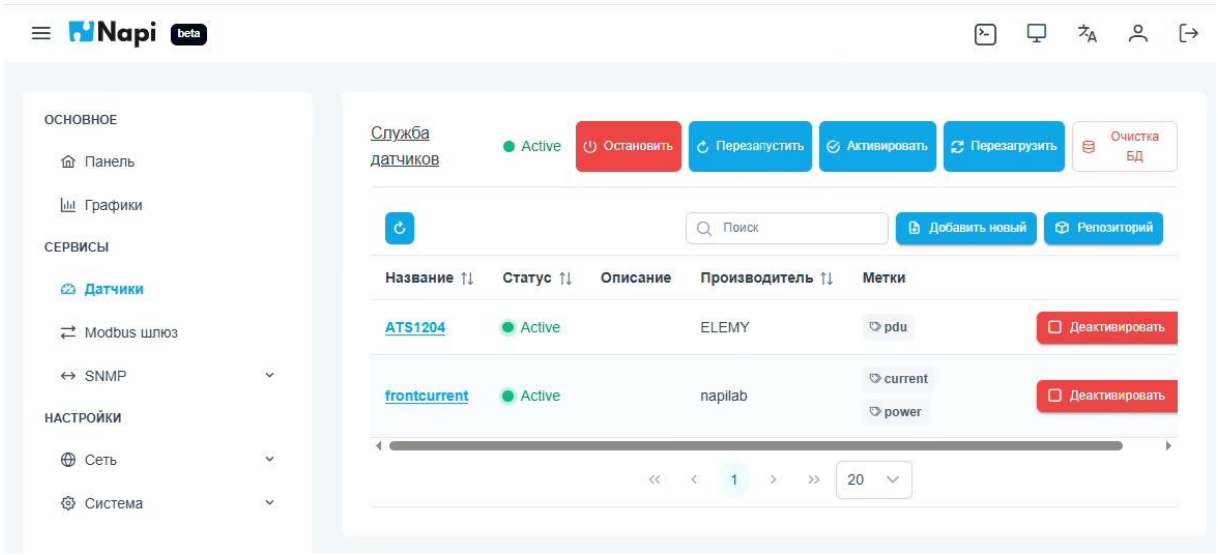

Рисунок 3 — Список датчиков и управление службой датчиков

12.4. Очистка базы данных датчиков

Функция очистки базы данных InfluxDB предназначена для очистки тестовой среды и возврата хранилища измерений к стартовому состоянию. Операция необратима и подтверждается через диалог с предупреждением.

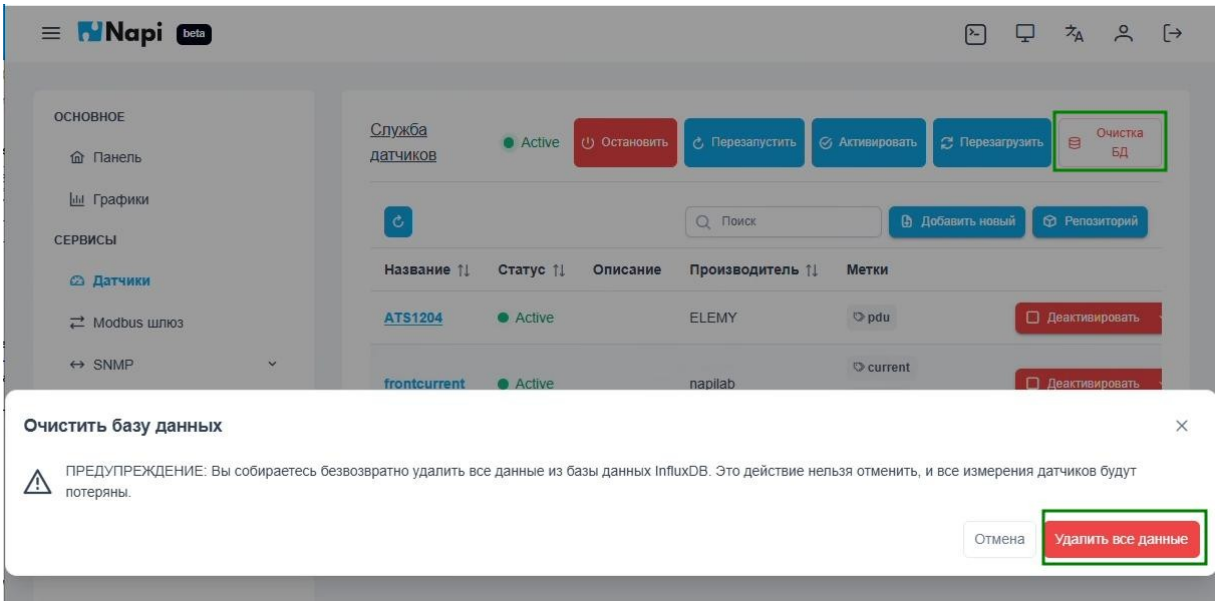

Рисунок 4 — Диалог очистки базы данных измерений

12.5. Редактирование конфигурации датчика

Экран редактирования позволяет менять параметры сбора: имя датчика, адрес устройства, последовательный порт, скорость обмена, список регистров и метрик. Конфигурация редактируется во встроенном редакторе в формате Telegraf (TOML).

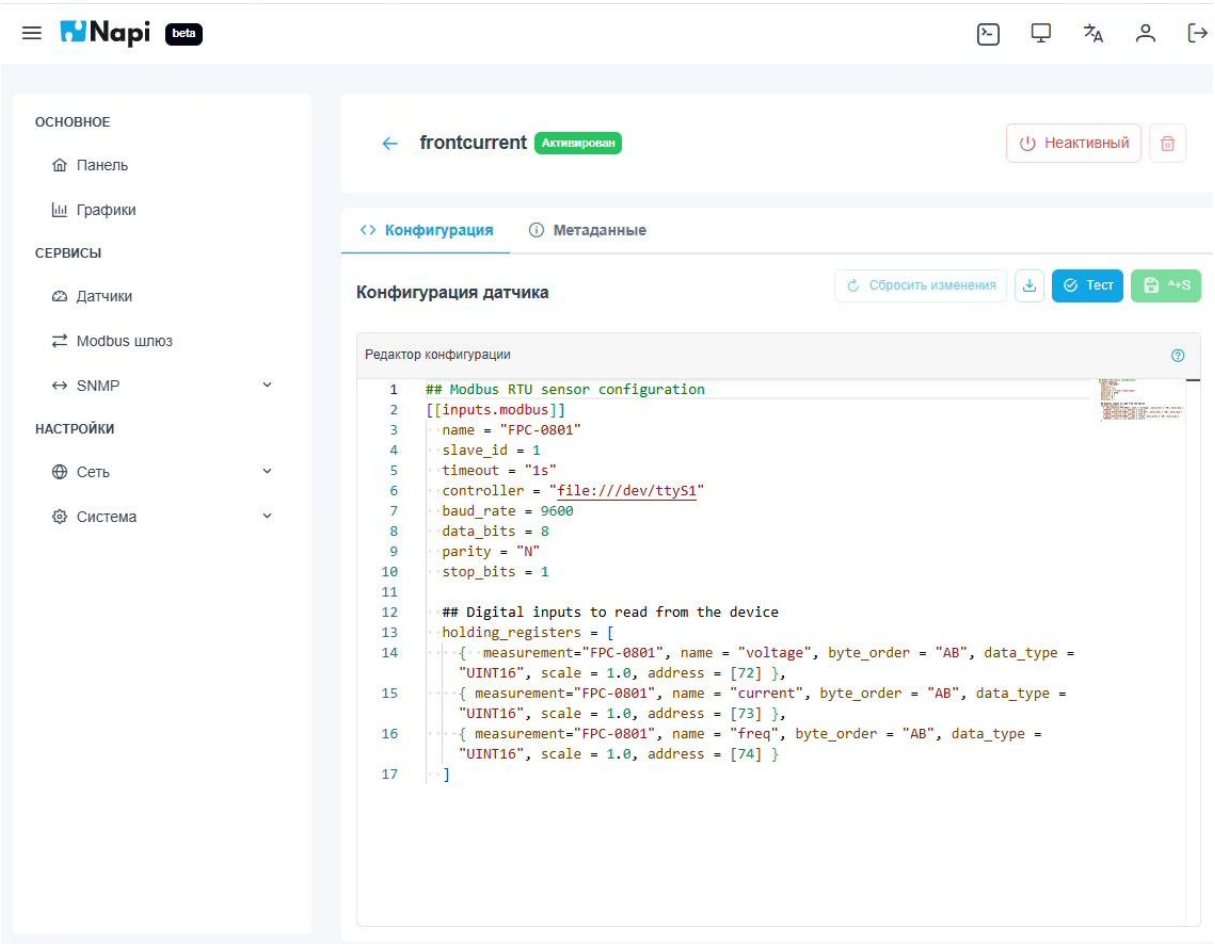

Рисунок 5 — Редактор конфигурации датчика Modbus RTU

12.6. Тестирование конфигурации датчика

Перед применением конфигурацию можно проверить «на лету»: система выполняет тестовый запуск Telegraf с указанной конфигурацией и показывает вывод измерений, полный лог запуска и код возврата.

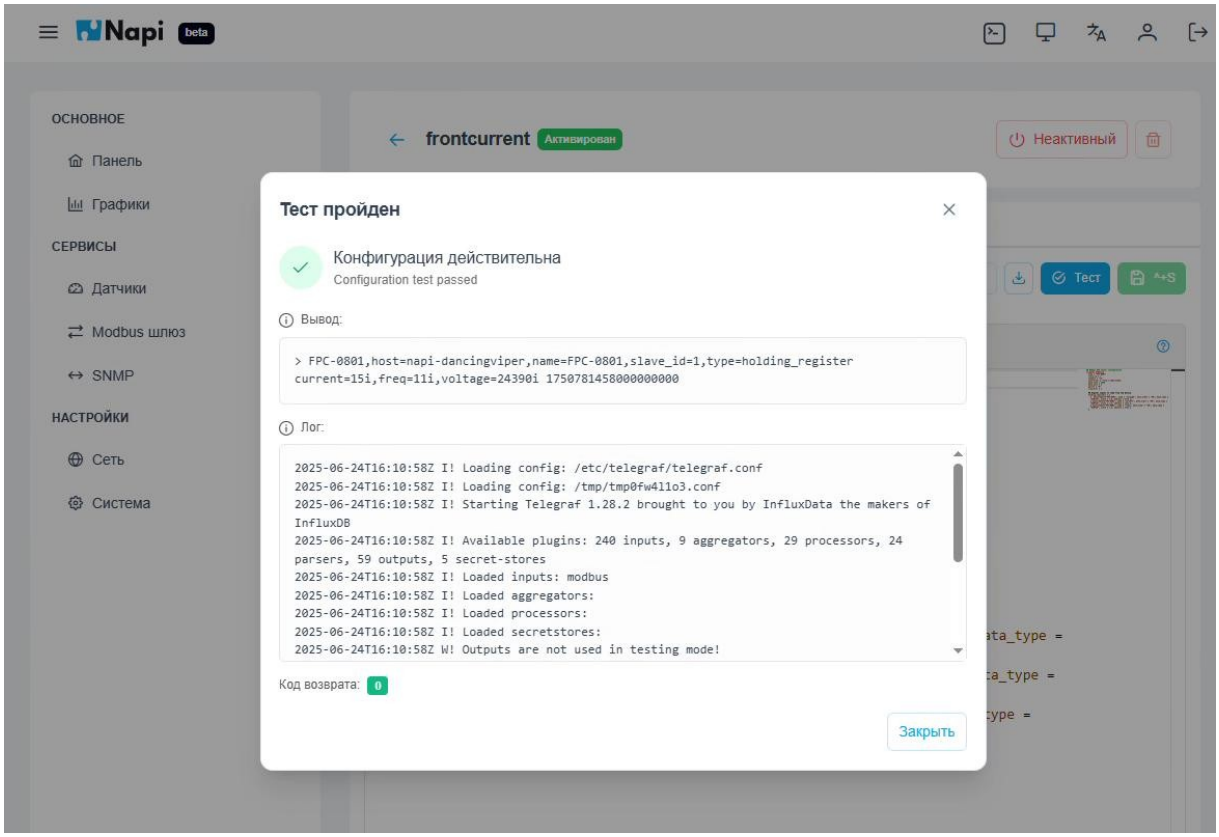

Рисунок 6 — Результат тестирования конфигурации датчика

12.7. Репозиторий датчиков

Экран репозитория позволяет выбирать готовые профили датчиков различных производителей (Elemy, Exagate, ICP DAS и др.), импортировать шаблоны конфигураций SNMP, Modbus RTU и Modbus TCP и расширять список поддерживаемых устройств.

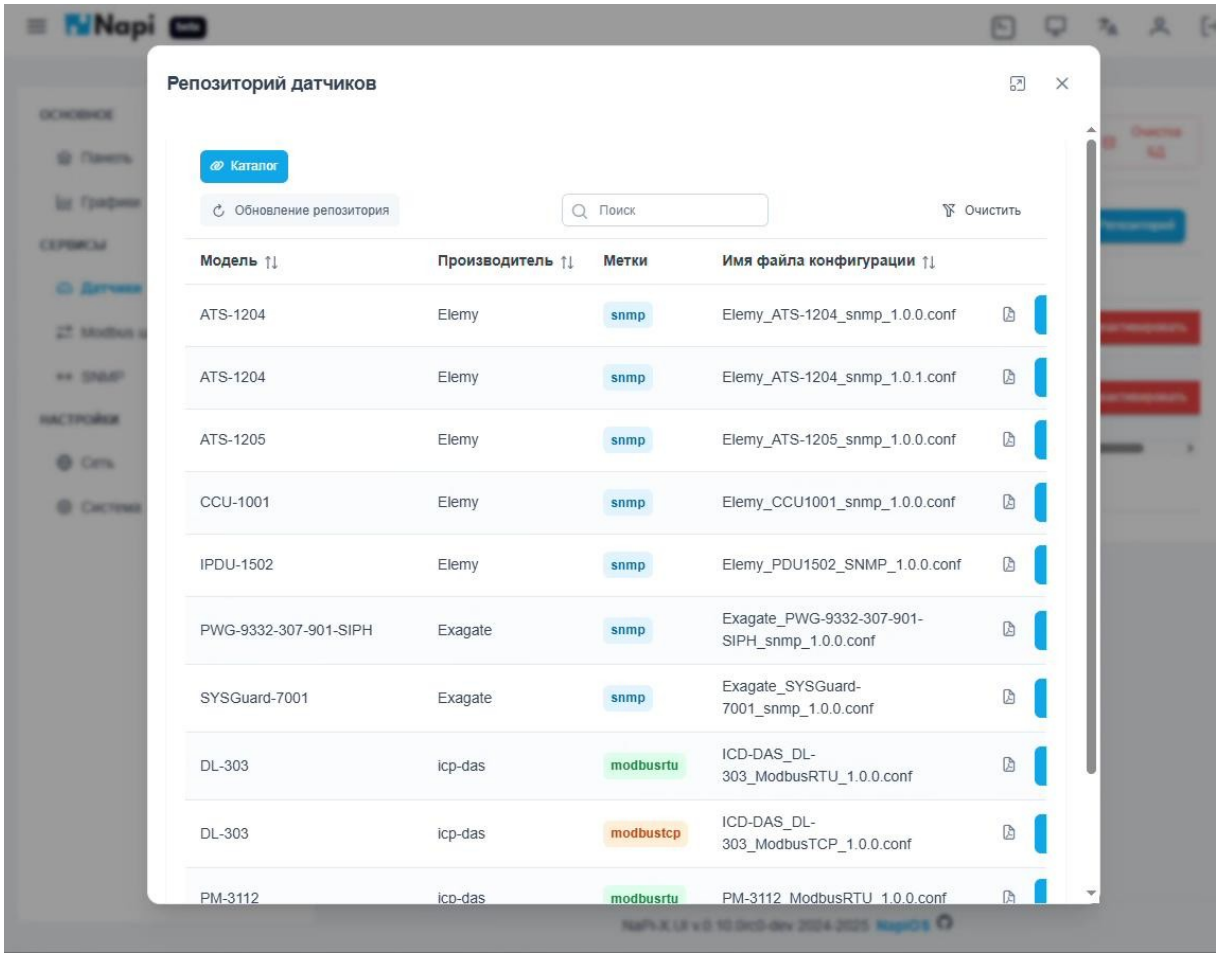

Рисунок 7 — Репозиторий готовых конфигураций датчиков

12.8. Графики значений датчика

Экран графиков демонстрирует значения регистров датчика во времени. Доступны выбор датчика, поля измерения и временного диапазона, экспорт данных и масштабирование по выделению.

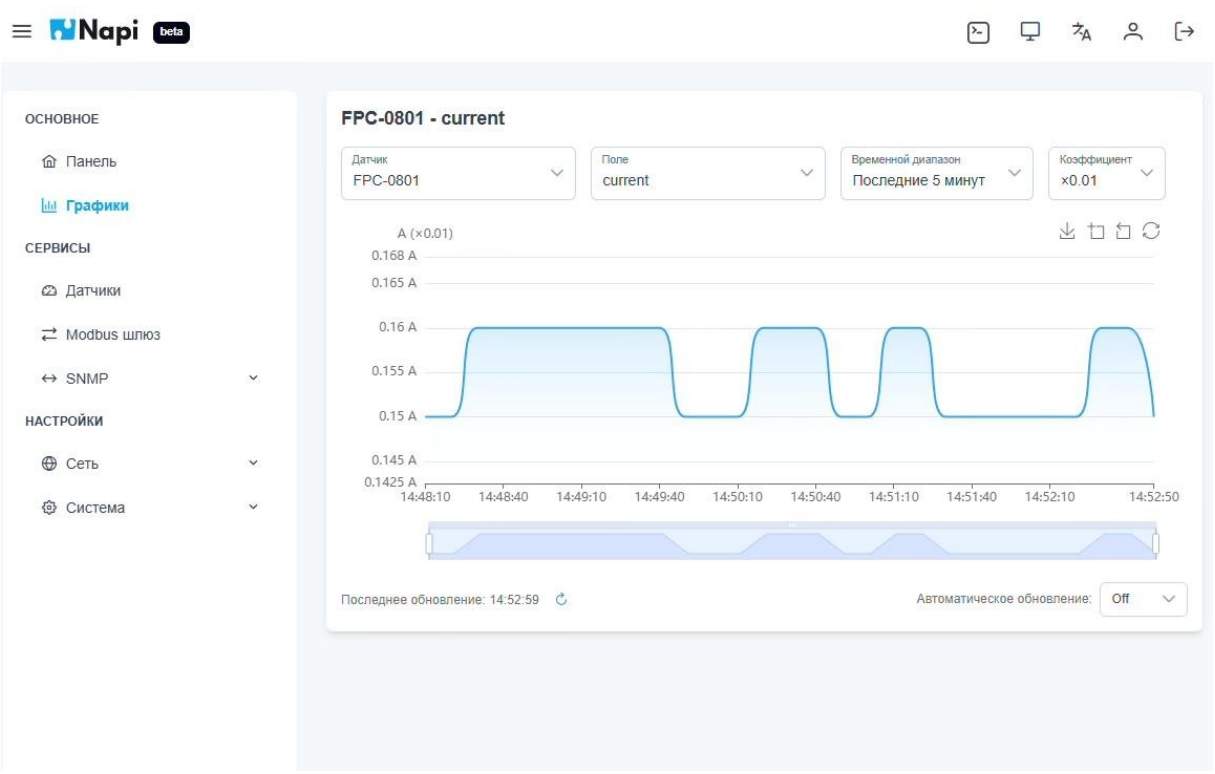

Рисунок 8 — График измеряемой величины датчика FPC-0801

12.9. Графики с множителем и изменением интервала

Отображаемые значения могут масштабироваться через коэффициент (множитель), также поддерживается изменение временного интервала отображения данных и навигация по мини-карте графика.

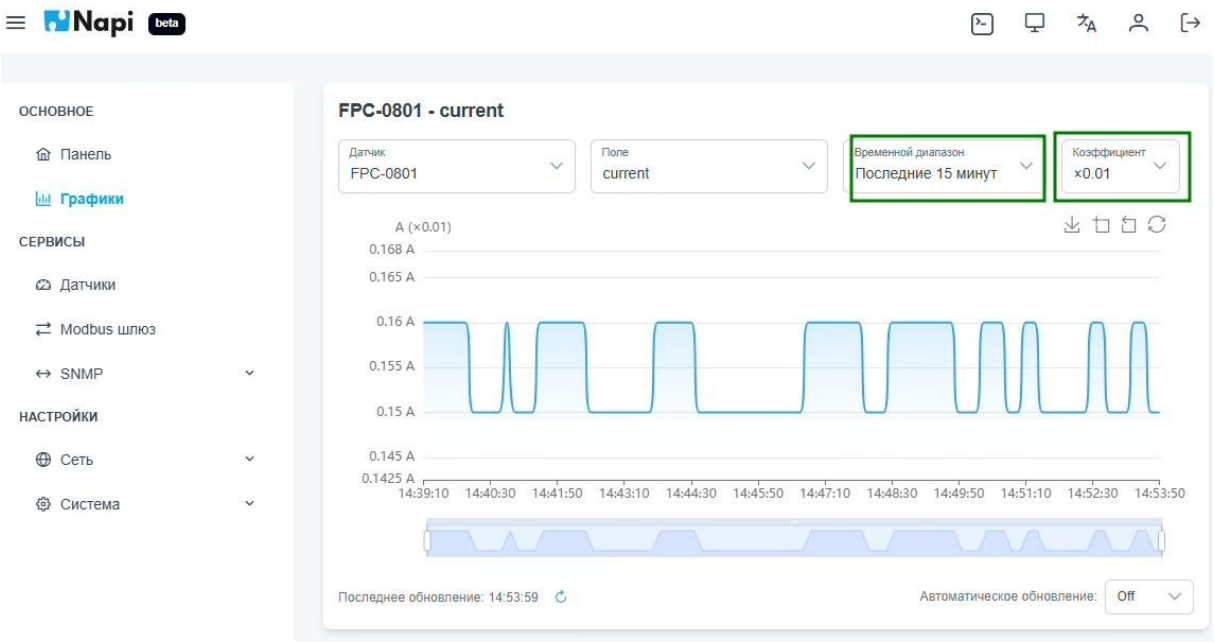

Рисунок 9 — График с коэффициентом ×0.01 и диапазоном 15 минут

12.10. Шлюз Modbus RTU — Modbus TCP

Экран шлюза Modbus позволяет настраивать преобразование Modbus RTU в Modbus TCP: параметры последовательного порта, тип управления направлением RS-485, адрес и порт TCP-сервера, лимиты соединений, тайм-ауты и параметры протокола. Отображается текущая сохраненная конфигурация и статус сервиса.

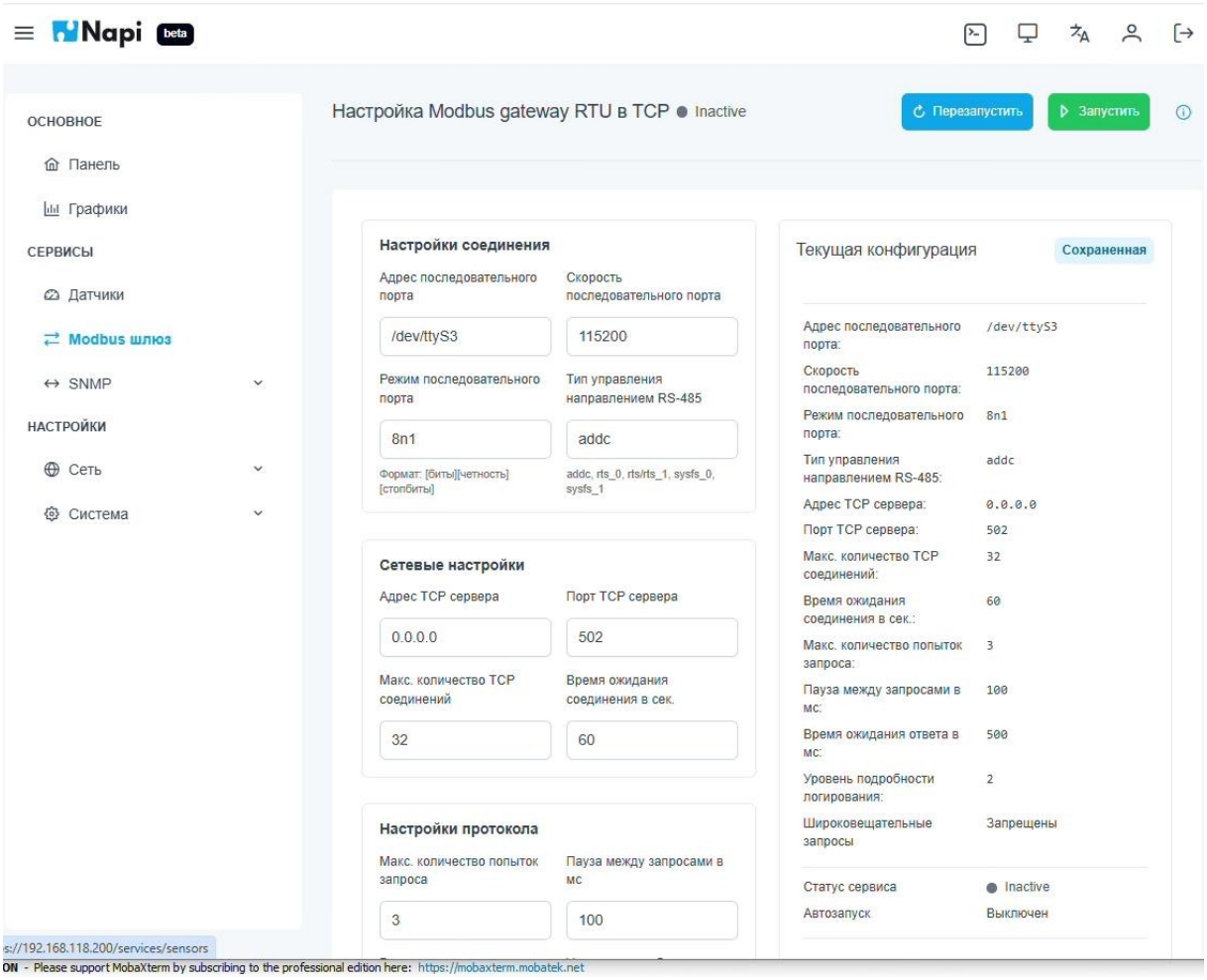

Рисунок 10 — Настройка шлюза Modbus RTU в TCP

12.11. Настройка даты и времени

Раздел настройки времени отображает текущее время, часовой пояс и статус NTP-синхронизации. Поддерживаются ручная установка времени, включение синхронизации NTP, управление списком NTP-серверов и работа с аппаратными часами RTC (синхронизация в обе стороны).

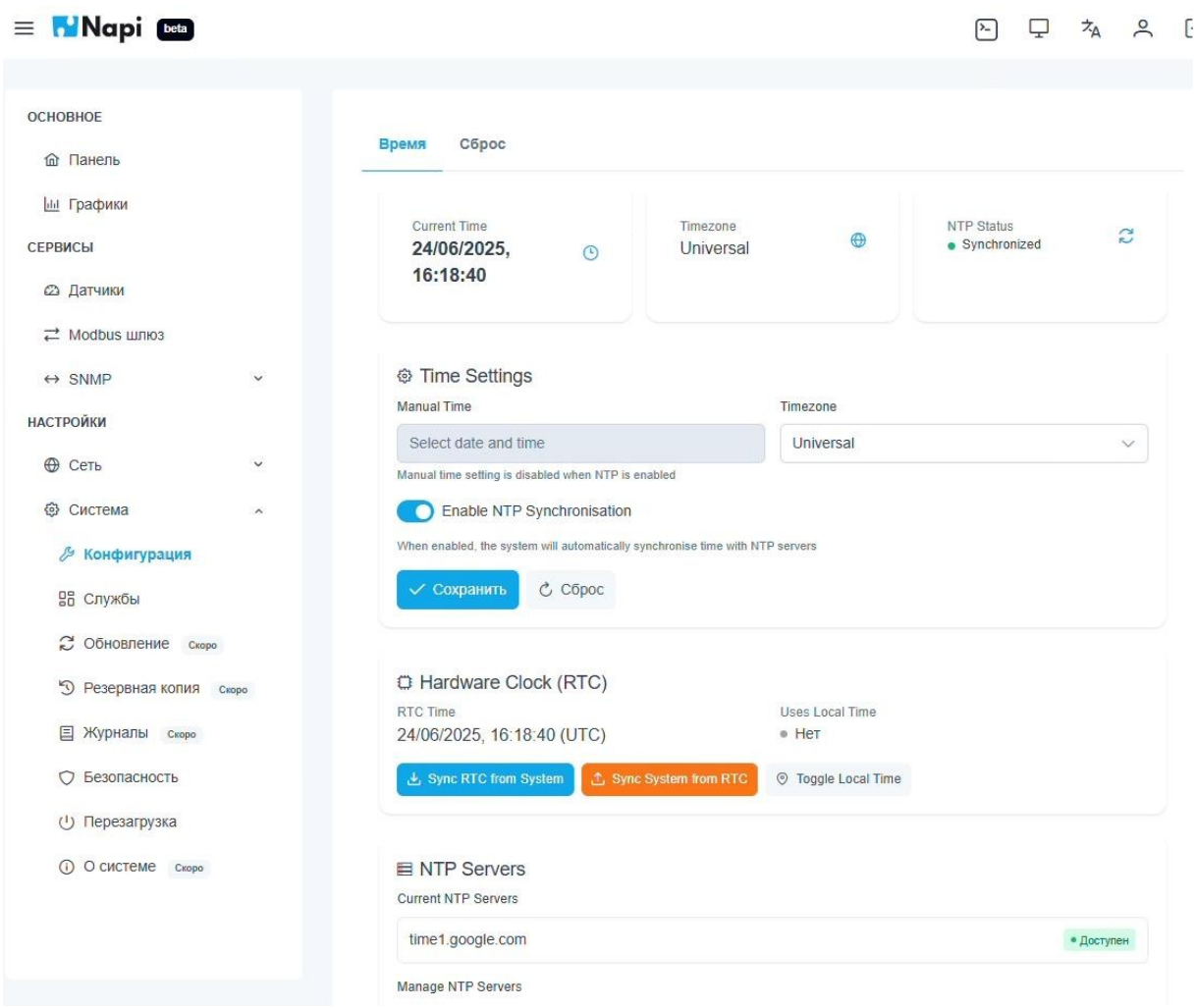

Рисунок 11 — Настройка даты, времени, NTP и RTC

12.12. Полный сброс системы

В разделе «Сброс» выполняется полный сброс устройства до заводских настроек: очистка пользовательского overlay, удаление конфигураций, ключей, сетевых настроек и журналов. Операция сопровождается критическим предупреждением и рекомендацией создать резервную копию.

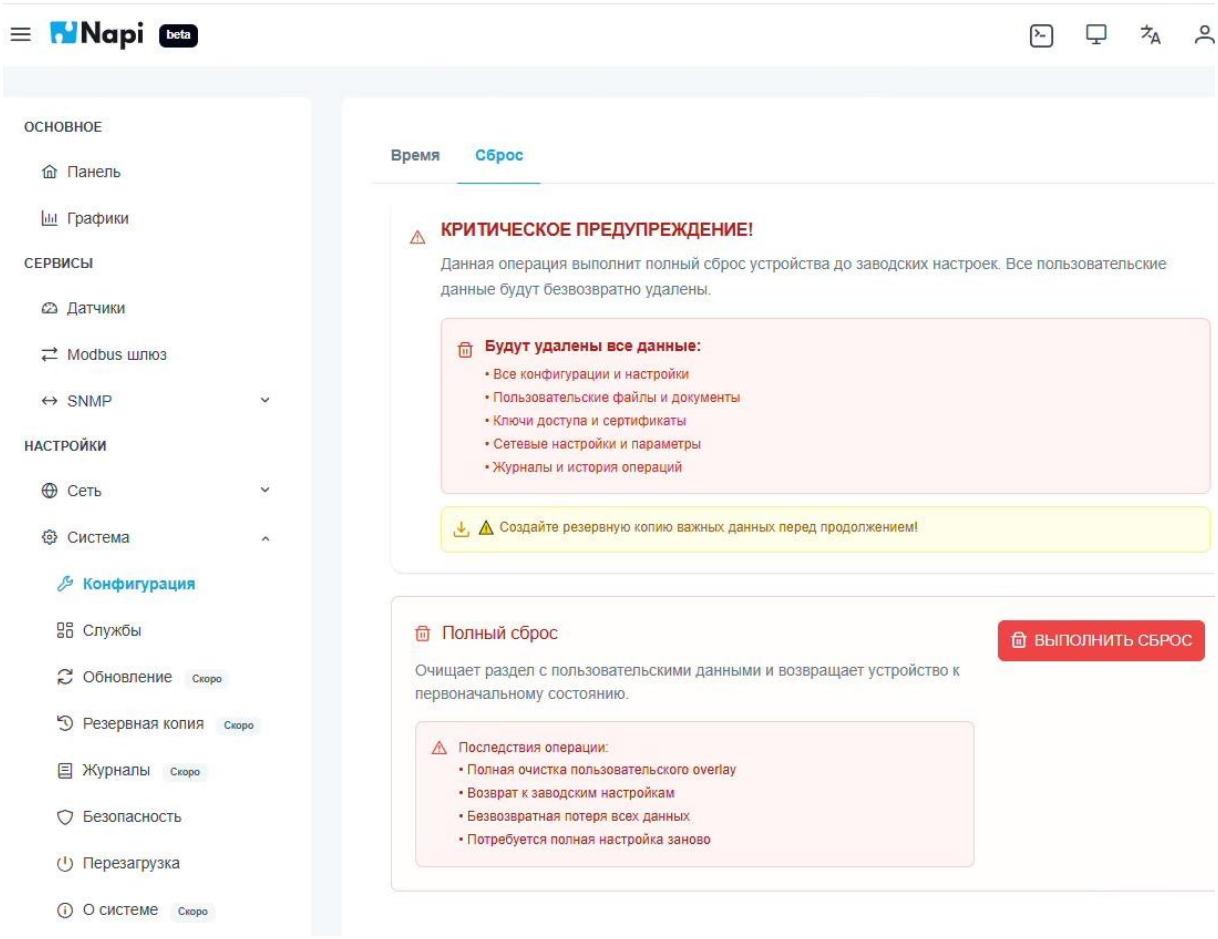

Рисунок 12 — Экран полного сброса до заводских настроек

12.13. Управление службами Linux

Экран управления службами показывает список systemd-сервисов, их описания, статусы и состояние активации. Поддерживаются поиск и фильтрация по имени, типу юнита и статусу.

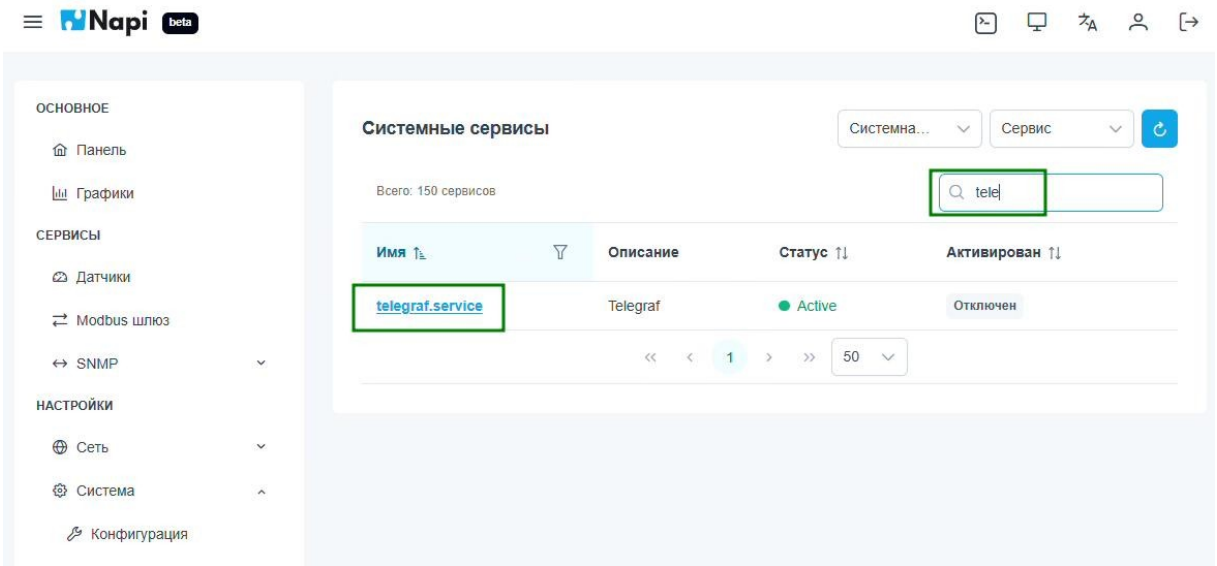

Рисунок 13 — Список системных сервисов с фильтрацией по имени

12.14. Карточка службы и просмотр логов

Карточка отдельной службы содержит состояние загрузки юнита, тип шины и юнита, информацию о выполнении (память, ЦП), команду запуска, пользователя, расположение unit-файла, а также блок последних логов с возможностью открыть полный журнал.

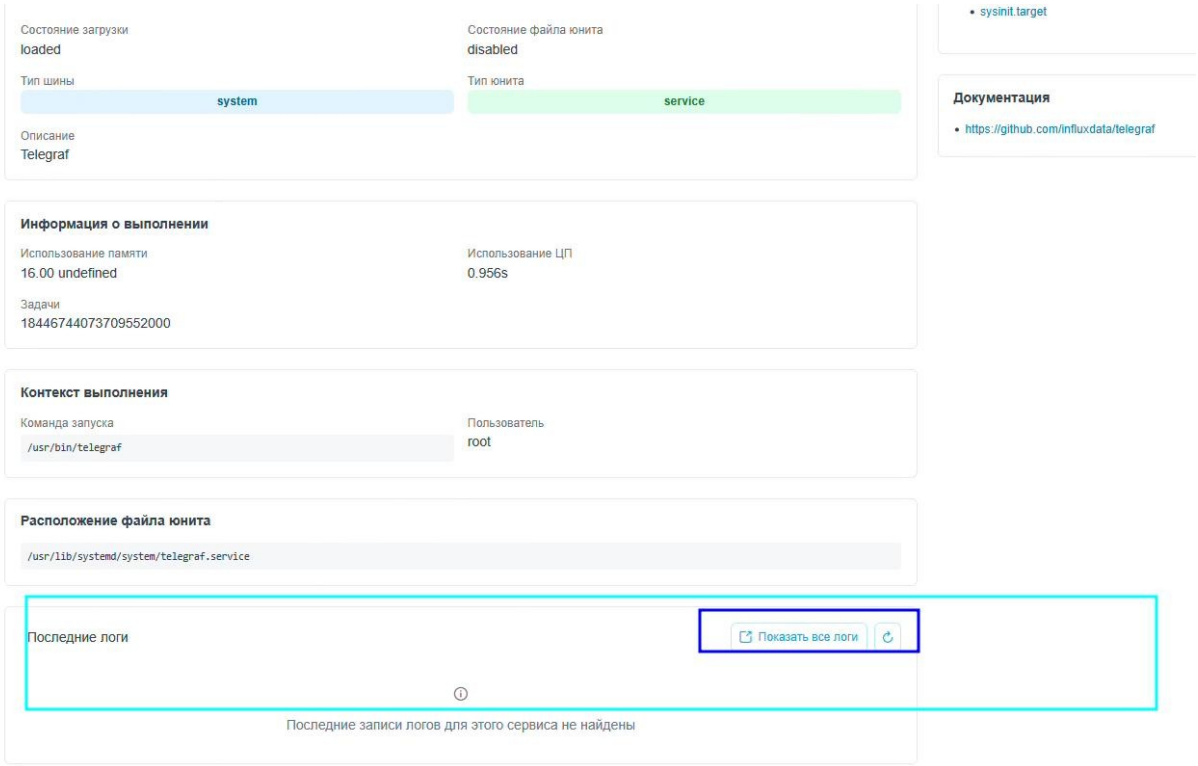

Рисунок 14 — Карточка службы telegraf.service с блоком логов

12.15. Веб-терминал

Веб-терминал обеспечивает удаленный доступ к командной строке через браузер (интеграция с TTYD). Доступ осуществляется по одноразовому токену, поддерживается подключение к целевому устройству по SSH.

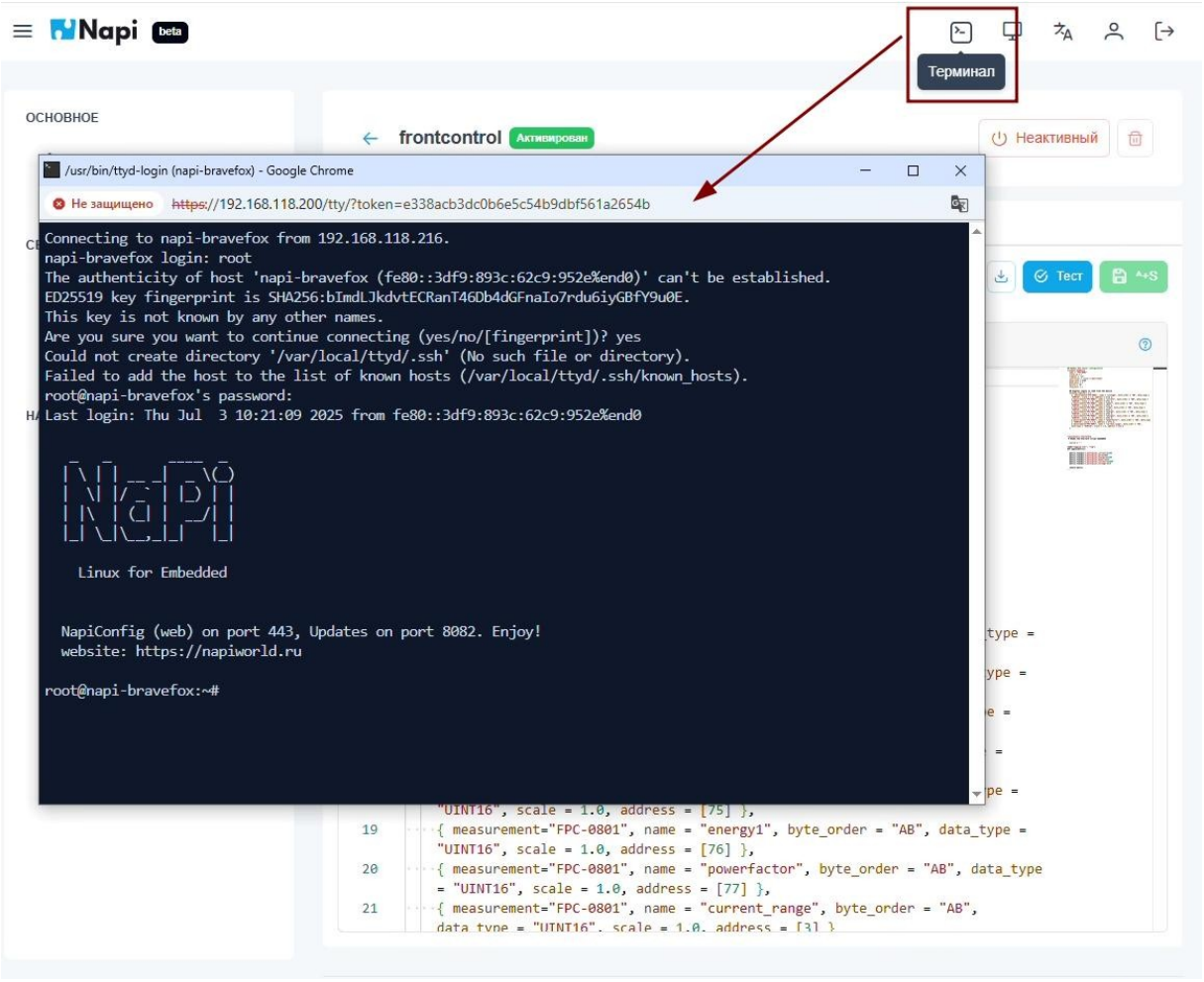

Рисунок 15 — Сессия веб-терминала с авторизацией по SSH

12.16. Мониторинг процессов в веб-терминале

В веб-терминале доступны полноэкранные консольные приложения; на рисунке показан запуск htop для мониторинга процессов, загрузки ЦП и памяти системы.

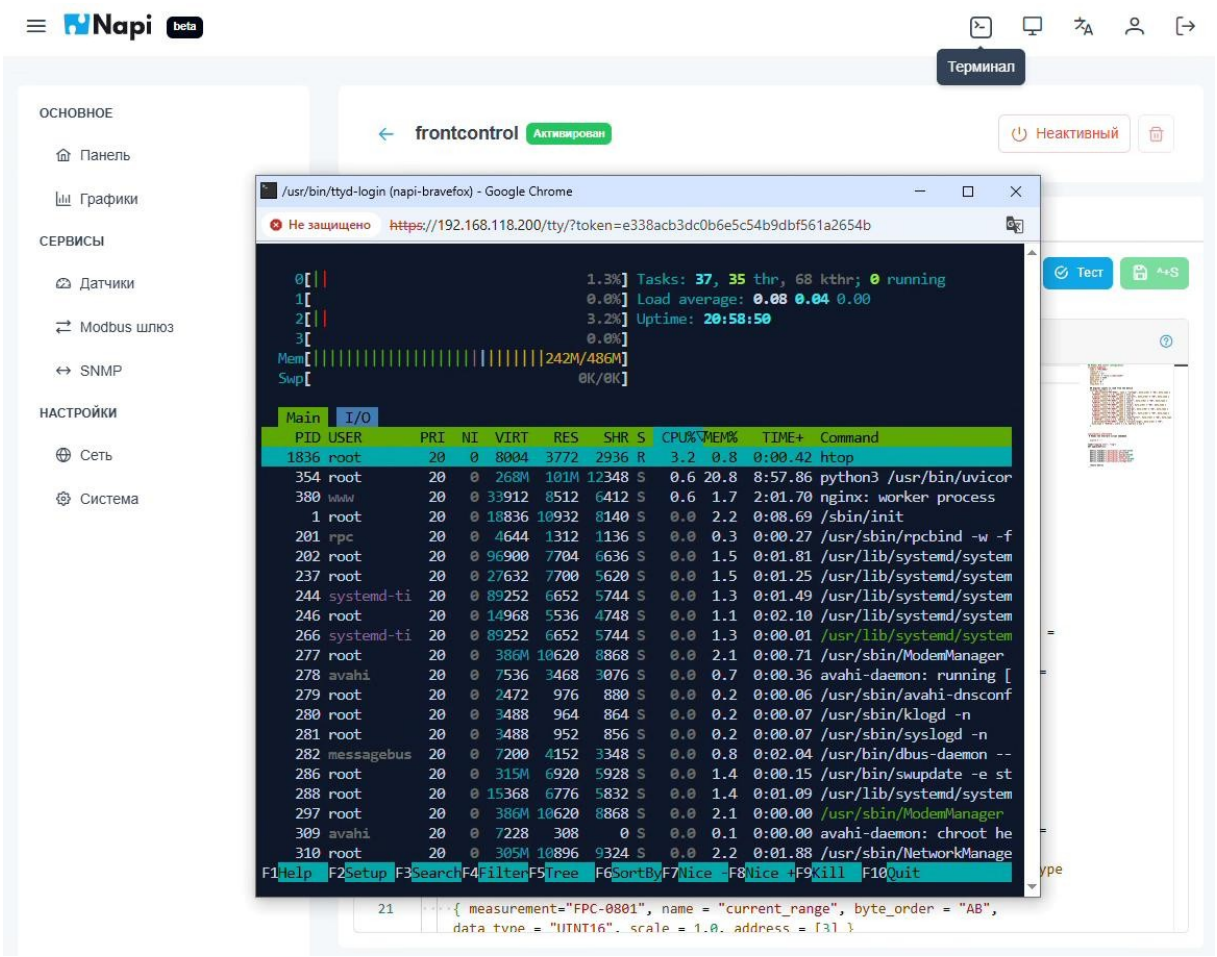

Рисунок 16 — htop, запущенный в веб-терминале