Описание функциональных характеристик программного обеспечения и информация, необходимая для установки и эксплуатации

Программный продукт «ABKInsight»

Дата: 2026-01-12

Основание: текущая версия репозиториев продукта в данном каталоге проекта (на дату формирования документа).

Состав продукта (репозитории):

- avk-scada-gateway (FastAPI/uvicorn): 010fdbb
- avk-scada-lan-server (worker внутри LAN): e0d7919
- avk-scada-frontend (SvelteKit/Svelte 5): e140c2f
- avk-scada-app (Python/pywebview desktop): e88a7f3
- avk-scada-docker (Docker Compose пакет): 8f5e2f0

Оглавление

Введение

Документ предназначен для описания функциональных характеристик программного обеспечения «ABKInsight», а также информации, необходимой для установки и эксплуатации.

Продукт состоит из нескольких компонентов (репозиториев), которые развертываются в разных сетевых зонах:

- **в локальной сети здания (LAN)** — `avk-scada-lan-server` (доступ к Modbus-контроллерам);
- **в серверной/DMZ/облаке** — `avk-scada-gateway` (шлюз и API, аутентификация, телеметрия);
- **у пользователя** — `avk-scada-frontend` (веб-интерфейс) и/или `avk-scada-app` (desktop-контейнер).

Документ является самостоятельным и содержит описание архитектуры, протоколов взаимодействия, требований и порядка развертывания/эксплуатации без необходимости обращаться к внутренней документации репозиториев.

Термины и сокращения

- **ПО** — программное обеспечение.
- **SCADA** — класс систем диспетчеризации/мониторинга и управления.
- **PLC/контроллер** — промышленный контроллер (в т.ч. Schneider), предоставляющий регистры.
- **Modbus TCP** — протокол обмена с контроллерами.
- **LAN** — локальная сеть объекта (здания).
- **WS / WebSocket** — протокол обмена данными в реальном времени между клиентами и шлюзом.
- **HTTP API** — HTTP-интерфейс (REST-like) шлюза для авторизации/администрирования и получения истории.
- **`lan_token`** — идентификатор здания/объекта, по которому шлюз маршрутизирует запросы на нужный LAN-сервер.
- **`registers.json`** — allowlist регистров (что можно читать/писать) + метаданные (типы, enum, маркеры).
- **`building.json`** — структура здания/установок для UI + привязка элементов интерфейса к регистрам.
- **`enums.json`** — справочник перечислений (enum), используемый LAN-сервером при сборке `registers.json`.
- `outside_temperature_registers.json` — список регистров, куда LAN-сервер записывает расчётную наружную температуру (сценарий outside_temperature).

Функциональные характеристики

Цели и назначение

ПО «ABKInsight» предназначено для мониторинга и управления инженерными установками здания (вентиляция, отопление, увлажнение и связанные подсистемы) через чтение/запись регистров контроллеров по Modbus внутри локальной сети объекта, с возможностью удаленного доступа операторов/инженеров через WebSocket-шлюз и веб-интерфейс.

Ключевые функции

- **Мониторинг в реальном времени**: периодический опрос регистров с группировкой в блоки, кэширование и отправка изменений в интерфейс.
- **Удаленное управление**: запись в разрешенные регистры (уставки, переключатели режимов, сброс аварий и т.п.) с учетом роли пользователя.
- Локальные автоматизации: shutdown_on_fire, outside_temperature и work_schedule (работают в LAN-сервере на локальных настройках/расписаниях).
- **Конфигурация “как данные”**: описание структуры здания и регистров в JSON (`building.json`, `registers.json`) с доставкой через шлюз и “горячими” обновлениями.
- **Аутентификация и разграничение доступа**: сессии, роли, ограничение на набор доступных зданий (`lan_token`).
- **Аварии и предупреждения**:
    - фиксация начала/окончания аварий по регистрам типа `Alarm`/`Warning`;
    - просмотр активных/архивных аварий;
    - (опционально) уведомления в Telegram.
- **История и графики**: запись телеметрии в ClickHouse и выдача данных для графиков/аналитики.
- **Администрирование**: управление пользователями, зданиями (доступами), просмотр сессий, история действий (для admin/super_admin).
- **Статус инфраструктуры**: фиксация онлайн/оффлайн/heartbeat состояния LAN-воркеров в ClickHouse.

Архитектура и потоки данных

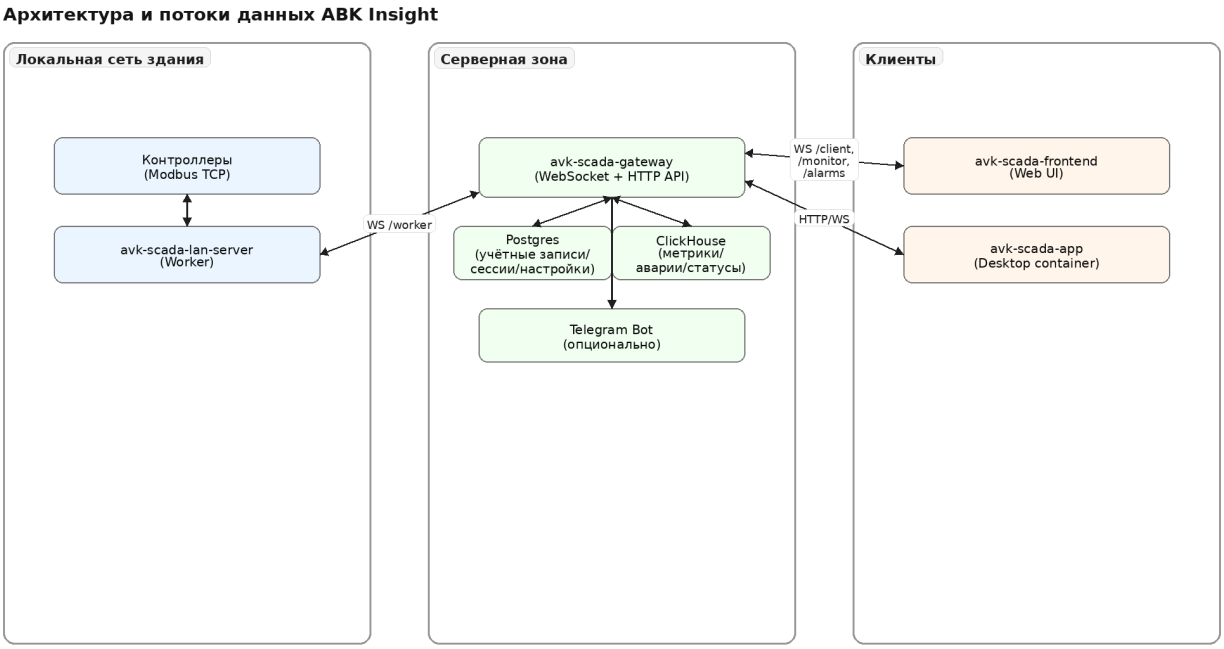


Рисунок — Архитектура и потоки данных ABKInsight

Ключевые потоки:

- **LAN-сервер ↔ контроллеры**: Modbus TCP (чтение/запись регистров).
- LAN-сервер ↔ шлюз: WebSocket /worker (задачи на запись, подписки мониторинга, передача конфигов/настроек/расписаний, телеметрия).
- **Клиент ↔ шлюз**:
  - WebSocket `/client` — запросы конфигов и операции записи;
  - WebSocket `/monitor` — подписка на чтение по интервалу;
  - WebSocket `/alarms` — live-поток аварий;
  - HTTP /api/* — авторизация, администрирование, настройки зданий/установок и сервисные эндпоинты (обновления воркеров и т.п.).
  - HTTP `/api/alarms`, `/api/charts/registers`, `/api/actions_history` — история аварий/графики/действия.

Важное ограничение текущей версии: клиентские приложения (web/desktop) получают конфиги и данные **только через шлюз**, поэтому для штатной работы требуется стабильная связность клиента со шлюзом (обычно — постоянное интернет-подключение или иной канал связи до сервера, где размещён gateway).

Критичные автоматизации выполняются локально в LAN-сервере и используют настройки, хранимые в его SQLite (LAN_DB_PATH), поэтому не зависят от внешнего канала связи.

План развития: предусматривается реализация fallback-механизма для временной работы **внутри одной локальной сети**, когда клиентское приложение находится в той же LAN, где доступен `avk-scada-lan-server` (в таком режиме клиент сможет продолжать работу при временной недоступности внешнего канала).

Протокол взаимодействия (WebSocket/HTTP)

Компоненты ABKInsight взаимодействуют по WebSocket и HTTP. Основной обмен "живыми" данными идёт по WebSocket-соединениям, кадры — текстовый JSON в UTF-8.

WebSocket-эндпоинты шлюза

- `/worker` — подключение LAN-серверов (воркеров) к шлюзу. В заголовках подключения передаются:
  - `X-Lan-Token` — идентификатор здания (обязателен);
  - `X-Worker-Token` — общий секрет (опционально, рекомендуется).
- `/client` — подключение клиентов (UI/desktop) для:
  - получения конфигов registers.json/building.json/outside_temperature_registers.json (запрос file_request);
  - операций записи в регистры (запрос `register_request`, `action=write`).
- `/monitor` — отдельное подключение клиентов (UI/desktop) для подписки на чтение по интервалу (запрос `watch_request`). В текущей схеме чтение не выполняется через `/client`, чтобы отделить постоянный поток мониторинга от операций записи.
- `/alarms` — live-поток событий аварий (стрим обновлений активных/закрытых аварий).

HTTP API шлюза

HTTP API предназначен для авторизации, администрирования и получения истории:

/api/auth/* — логин/логаут, профиль; device-flow для desktop: /api/auth/device/* и /oauth/device;

/api/auth/sessions, /api/auth/sessions/revoke, /api/auth/sessions/revoke_all — управление сессиями (admin/super_admin);

/api/users, /api/users/{id}/buildings, /api/buildings, /api/lan_tokens/live — управление пользователями и доступом к зданиям;

/api/building-settings/{lan_token}/{key} — чтение/изменение настроек здания (прокси на LAN-сервер);

/api/machine-settings/{lan_token} и /api/machine-settings/{lan_token}/{machine_name}/work-schedule — настройки установок и расписаний;

/api/alarms — история аварий (если включён ClickHouse);

/api/charts/registers — данные для графиков (если включён ClickHouse);

/api/actions_history — история действий (для ролей admin/super_admin, если включён ClickHouse);

/api/dev/lan/update, /api/lan/update — загрузка и раздача обновлений LAN-сервера.

Аутентификация (WS и HTTP)

- **Web-сессия**: шлюз выдаёт `session_token`, который используется как cookie `avk_session_token=<session_token>` (в браузере).
- **Desktop-сессия**: токен передаётся в query-параметре `session_token` (используется, когда cookie-сценарий недоступен).
- При неуспешной аутентификации WebSocket-соединение закрывается кодом `4401`.

Маршрутизация по зданиям (`lan_token`)

- Все клиентские запросы к WebSocket-шлюзу содержат `lan_token`.
- По `lan_token` шлюз выбирает, на какой подключенный LAN-воркер пересылать операции/подписки.
- При отсутствии воркера возвращается ошибка (например, `GW_LAN_OFFLINE`), при запрете доступа — `GW_FORBIDDEN`.

Ключевые сообщения (кратко)

- file_request → file_response/file_update: получение/обновление registers.json, building.json и outside_temperature_registers.json через шлюз.
- `watch_request` (на `/monitor`) → поток `register_response`: мониторинговые чтения по интервалу; в ответах приходят только изменившиеся значения (и первичный “снимок”).
- `register_request` (на `/client`) → `register_response`: операции записи в регистры (роль `operator` писать не может).
- `gateway_error`: унифицированная ошибка шлюза (с текстом и `error_id`).

- building_setting_update: обновление значения настройки здания (рассылка клиентам с доступом к lan_token).
- machine_settings_update: обновление настроек установки (room_machines/заметки) для admin/super_admin.

Функциональные модули

1) `avk-scada-lan-server` (LAN worker)

Назначение: выполнение операций чтения/записи по Modbus внутри локальной сети здания и постоянное соединение со шлюзом.

Основные функции:

- подключение к шлюзу на `/worker` с заголовком `X-Lan-Token` (идентификация здания) и опциональным `X-Worker-Token` (общий секрет);
- **очередь задач**: операции записи выполняются строго последовательно (без параллельной нагрузки на контроллер);
- **мониторинг**: регистры `mode=read` группируются по контроллеру/функции в блоки, читаются циклически, значения кэшируются; при изменениях отправляются обновления;
- **allowlist**: чтение/запись возможны только для регистров, описанных в `registers.json`;
- публикация конфигов: отдаёт registers.json, building.json и outside_temperature_registers.json через шлюз по file_request; при изменениях файлов шлёт file_update;
- локальные автоматизации: fire_shutdown, outside_temperature, work_schedule; настройки зданий и расписания хранятся в локальной SQLite (LAN_DB_PATH), поэтому критичные сценарии не зависят от внешнего канала.

Ключевые переменные окружения:

- `LAN_GATEWAY_URL` (пример: `ws://<gateway-host>:8765/worker`)
- `LAN_TOKEN` (токен здания)
- `LAN_GATEWAY_TOKEN` (если используется общий секрет шлюза)
- конфиги и БД: LAN_CONFIG_DIR, LAN_DB_PATH
- лимиты и соединения: LAN_MAX_MESSAGE_SIZE, LAN_RECONNECT_DELAY, LAN_WORKER_ID
- тайминги/мониторинг: LAN_CONTROLLER_TIMEOUT, LAN_IDLE_SECONDS, LAN_MONITOR_MAX_BLOCK_SIZE
- файлы и телеметрия: LAN_FILE_WATCH_INTERVAL, LAN_LOG_FILE, LAN_TELEMETRY_*, LAN_HEARTBEAT_INTERVAL

- автоматизации: LAN_AUTOMATION_FIRE_SHUTDOWN_ENABLED, LAN_AUTOMATION_OUTSIDE_TEMPERATURE_ENABLED, LAN_AUTOMATION_OUTSIDE_TEMPERATURE_WRITE_INTERVAL, LAN_AUTOMATION_WORK_SCHEDULE_ENABLED

2) `avk-scada-gateway` (WebSocket-шлюз + API)

Назначение: единая точка доступа для клиентов (UI/desktop) и маршрутизация запросов на нужный LAN-сервер, а также авторизация, хранение и выдача истории.

Основные функции:

- **WebSocket-шлюзирование и маршрутизация** по `lan_token`:
  - `/worker` — подключение LAN-серверов;
  - `/client` — конфиги + операции записи (write);
  - `/monitor` — подписка на чтение;
  - `/alarms` — live-поток аварий.
- **HTTP API** (префикс `/api`):
  - авторизация и сессии, device-flow для desktop (`/api/auth/*`, `/oauth/device`);
  - управление пользователями/правами/зданиями (/api/users, /api/buildings, /api/lan_tokens/live) и сессиями (/api/auth/sessions/*);
  - настройки зданий и установок (/api/building-settings/{lan_token}/{key}, /api/machine-settings/*), включая рассылку building_setting_update и machine_settings_update по WS.
- **Интеграции хранения**:
  - Postgres — пользователи/сессии/права доступа, настройки установок (room_machines/заметки), привязки Telegram, журнал действий.
- ClickHouse — телеметрия регистров, аварии, статус воркеров, история действий.
- **Telegram-уведомления** (опционально): рассылка начала/окончания аварий привязанным пользователям.
- Обновления LAN-серверов: /api/dev/lan/update → lan_update_available → /api/lan/update.

Ключевые переменные окружения:

- WS/HTTP: `GATEWAY_HOST`, `GATEWAY_PORT`, `GATEWAY_CLIENT_PATH`, `GATEWAY_WORKER_PATH`, `GATEWAY_MONITOR_PATH`

- обновления воркеров: GATEWAY_DEV_KEY, GATEWAY_DEV_UPDATE_API_PATH, GATEWAY_WORKER_UPDATE_API_PATH, GATEWAY_UPDATE_DIR
- безопасность: `GATEWAY_WORKER_TOKEN`, `GATEWAY_CORS_ORIGINS`
- Postgres: DATABASE_URL или POSTGRES_HOST/POSTGRES_PORT/POSTGRES_DB/POSTGRES_USER/POSTGRES_PASSWORD
- ClickHouse: CLICKHOUSE_ENABLED, CLICKHOUSE_HOST, CLICKHOUSE_PORT, CLICKHOUSE_DATABASE, CLICKHOUSE_USERNAME, CLICKHOUSE_PASSWORD, CLICKHOUSE_CREATE_TABLES, …
- авторизация: AUTH_DEFAULT_ADMIN_LOGIN, AUTH_DEFAULT_ADMIN_PASSWORD, AUTH_SESSION_TTL, AUTH_SESSION_MAX_IDLE, AUTH_LOGIN_RATE_LIMIT, AUTH_PASSWORD_SCHEME
- Telegram: `TELEGRAM_BOT_TOKEN`

Telegram-уведомления об авариях

Шлюз может отправлять уведомления в Telegram о **начале** и **окончании** аварий (с учётом доступа пользователя к `lan_token`).

Условия работы:

- Telegram-интеграция включается при заданной переменной `TELEGRAM_BOT_TOKEN`.
- Для формирования событий аварий требуется включённая телеметрия (CLICKHOUSE_ENABLED=1), так как аварии вычисляются из потока мониторинговых/телеметрических данных.
- Шлюзу требуется исходящий доступ в интернет к Telegram API.

Привязка Telegram к учётной записи:

- пользователь пишет боту в личный чат и выполняет привязку;
- поддерживаются команды:
  - `/link` — привязать Telegram к пользователю (в процессе бот запрашивает логин/пароль);
  - `/unlink` — отвязать Telegram.
- привязка действует бессрочно и автоматически отзывается при смене пароля пользователя (или по команде `/unlink`).

3) `avk-scada-frontend` (Web UI)

Назначение: пользовательский интерфейс для мониторинга/управления установками, отображающий структуру здания и “живые” значения регистров.

Основные функции UI:

- `/auth` — авторизация (web: логин/пароль; desktop: device-flow).
- `/building` — обзор здания/зон и быстрые панели оперативного контроля.
- `/machine` — экран установки: холст устройств, уставки/переключатели, сброс аварий, графики.
- `/emergencies` — аварии и архив: история по HTTP + live-поток по WS, фильтры/поиск.
- `/admin_dashboard` — администрирование (для `admin`/`super_admin`).

Взаимодействие со шлюзом:

- получает `registers.json` и `building.json` по `file_request` через `WS /client` и подписывается на обновления (`file_update`);
- запускает `watch_request` на `WS /monitor` для чтения по интервалу;
- отправляет операции записи по `WS /client` (`register_request`, action=`write`).

Ключевые переменные окружения (Vite):

- `VITE_API_BASE_URL` (база для `/api/*`)
- `VITE_GATEWAY_CLIENT_URL` (WS `/client`)
- `VITE_GATEWAY_MONITOR_URL` (WS `/monitor`)
- опционально: VITE_FORCE_DEVICE_AUTH_FLOW, VITE_GATEWAY_ALARMS_HTTP_URL, VITE_GATEWAY_ALARMS_WS_URL, VITE_GATEWAY_CHARTS_HTTP_URL

4) `avk-scada-app` (Desktop container)

Назначение: десктопное приложение на Python/pywebview для запуска интерфейса в отдельном окне (актуально для рабочих мест без “полноценного” браузерного сценария или для упрощения доступа).

Функции:

- упаковка в исполняемый файл (PyInstaller);
- запуск окна приложения (pywebview).

Примечание по текущей реализации: URL, который открывает окно, задан в `avk-scada-app/main.py` (на момент документа — `https://perenatal.avk-vozduhovod.ru/`). При необходимости автономной работы со статической сборкой фронтенда требуется адаптация логики открытия URL на локальный HTTP-сервер и/или использование содержимого каталога `avk-scada-app/web`.

Роли и права доступа

Роли задаются в шлюзе (`avk-scada-gateway`) и используются для ограничения операций и доступов к зданиям.

Доступные роли:

- `operator` — оператор (только чтение; запись запрещена).
- `engineer` — инженер (доступ к чтению/записи в рамках разрешенных зданий).
- admin — администратор (управление пользователями/доступами в рамках своей зоны ответственности; доступ к настройкам зданий/установок и расписаниям, требующим admin).
- `super_admin` — супер-админ (доступ ко всем зданиям и административным функциям).

Ограничения по зданиям:

- для большинства пользователей доступ ограничен списком `lan_token`, выданным администратором;
- `super_admin` может работать со всеми активными `lan_token` (включая список подключенных воркеров).

Конфигурация (building/registers/enums)

`registers.json`

Расположение (по умолчанию):
`avk-scada-lan-server/config/registers.json` (путь переопределяется `LAN_CONFIG_DIR`).

Назначение:

- allowlist регистров (что именно можно читать/писать);
- описание типа данных/декодирования (`data_type`, `word_order`, `scaling_factor`, `bit_index`, …);
- признаки аварийности (`type: "Alarm" | "Warning"`), enum-описания, служебные маркеры в `note` для UI.

Обязательные для корректной работы поля зависят от типа регистра, но типовой набор включает:

- координаты контроллера: `moxa_ip`/`moxa_port` (или `ip`/`port`), `slave_id`;
- адресация/декодирование: `register_address`, `register_count`, `data_type`, `read_function`, `word_order`, `connection_type`;
- режим: mode (read, write) + write_function для записываемых регистров;
- метаданные: `machine`, `name`, `note`, `type`, `enum`.

`building.json`

Расположение (по умолчанию):
`avk-scada-lan-server/config/building.json`.

Назначение:

- структура здания: группы этажей/зон, список установок (machines), источники наружной температуры (outside_temperatures);
- список устройств на экране установки и привязки UI-ключей (props/settings) к регистрам; props.status и settings.start_stop используются автоматизацией расписаний;
- списки аварий/сбросов аварий (используются UI и автоматизациями).

Принцип привязки:

- многие ссылки — это просто `register_address` (число), которое резолвится в контексте машины по паре `(machine, register_address)`;
- в местах без контекста машины допускаются ссылки строкой формата `{ip}:{port}:{address}`.

`enums.json`

Расположение (по умолчанию): `avk-scada-lan-server/config/enums.json`.

Назначение:

- справочник перечислений (enum) для удобного переиспользования в `registers.json` по имени;
- наружу не публикуется напрямую, но при выдаче `registers.json` LAN-сервер “разворачивает” enum в объект `{name, description?, values}`.
- `outside_temperature_registers.json`
- Расположение (по умолчанию): avk-scada-lan-server/config/outside_temperature_registers.json.

- Назначение: список регистров, куда LAN-сервер записывает расчётную наружную температуру (сценарий outside_temperature); используется при включённой автоматизации.

Хранение, телеметрия и журналирование

Postgres

Используется шлюзом для:

- хранения пользователей и ролей, привязок lan_token;
- управления сессиями (`web`/`desktop`);
- настроек установок (room_machines/заметки) и журнала действий;
- привязки Telegram-сессий (если включен бот).
- LAN-сервер (SQLite, LAN_DB_PATH)
- Используется для: хранения настроек зданий (shutdown_on_fire, outside_temperature_mode) и расписаний работы установок.

ClickHouse

Используется шлюзом для:

- записи телеметрии регистров (таблица метрик);
- фиксации аварий (таблица событий аварий);
- фиксации состояния воркеров (online/offline/heartbeat);
- выдачи данных для графиков и истории действий.

Включение/отключение: CLICKHOUSE_ENABLED=1/0.

Информация для установки и эксплуатации

Технические требования

Фактические требования зависят от количества регистров, частоты опроса и числа одновременных клиентов. Ниже приведены базовые ориентиры.

Серверная часть (шлюз + БД)

- ОС: Debian 13 (протестированная конфигурация) + Docker Compose; публикация наружу через Nginx + SSL (certbot).
- Ресурсы (минимально рекомендуемые): 2 CPU, 4–8 GB RAM, SSD-диск (зависит от объема телеметрии).
- ПО:
    - `avk-scada-gateway`: Python 3.13+ (или контейнер);
    - Postgres 17+;
    - ClickHouse 25+ (если нужна история/графики/архив аварий).

LAN-сервер

- ОС: Linux или Windows (поддерживается запуск как Windows Service).
- Ресурсы: 1–2 CPU, 1–2 GB RAM (зависит от нагрузки/таймаутов контроллеров).
- Доступ в сеть контроллеров Modbus TCP (обычно порт `502/tcp`) и исходящий доступ к шлюзу по WebSocket.

Клиентское рабочее место (web)

- Современный браузер с поддержкой WebSocket и JavaScript (Chrome/Edge/Firefox/Yandex).
- Рекомендуемое разрешение: от 1280×720.

Клиентское рабочее место (desktop)

- ОС: Windows (для сборки/использования `avk-scada-app` в виде `.exe`).
- Установка зависит от способа поставки (готовый `.exe` или сборка из исходников).

Сетевые требования и порты

Типовые порты (могут быть изменены настройками):

- `80/tcp`, `443/tcp` — Nginx (публикация интерфейса и шлюза на домены, SSL).
- `8765/tcp` — `avk-scada-gateway` (WS + HTTP API).
- `5173/tcp` — `avk-scada-frontend` в режиме `npm run prod` (если используется Node-сервер).
- `5432/tcp` — Postgres.
- `8123/tcp` и/или `9000/tcp` — ClickHouse (HTTP/native).

- `502/tcp` — Modbus TCP (контроллеры в LAN).

Важно: LAN-сервер инициирует исходящее подключение к шлюзу; входящих портов на LAN-сервере обычно не требуется.

Развертывание

См. документ «Руководство по развертыванию ABKInsight.docx».

Первоначальная настройка

1) Подготовка БД и “первого администратора”

При первом запуске шлюз может автоматически создать супер-админа, если таблица пользователей пуста:

- `AUTH_DEFAULT_ADMIN_LOGIN`
- `AUTH_DEFAULT_ADMIN_PASSWORD`

Рекомендуется сразу после запуска:

- выполнить вход под первичной учетной записью;
- создать постоянных пользователей и отключить/сменить пароль bootstrap-пользователя.

2) Регистрация здания (lan_token) и запуск воркера

- выберите/зафиксируйте `lan_token` для здания (строка);
- запустите на объекте `avk-scada-lan-server` с `LAN_TOKEN=<lan_token>`;
- назначьте доступ к этому `lan_token` пользователям через `avk-scada-gateway` (админка).

3) Настройка конфигов здания

- заполните `registers.json` реальными регистрами и отметьте:
  - режимы (`mode`), функции чтения/записи;
  - аварийные регистры (`type: Alarm/Warning`), если нужен архив аварий;
  - `enum`, если нужны человекочитаемые значения.
- заполните `building.json` структурой здания/установок и связями `props/settings`.
- при использовании автоматики наружной температуры подготовьте outside_temperature_registers.json (список регистров для записи температуры).

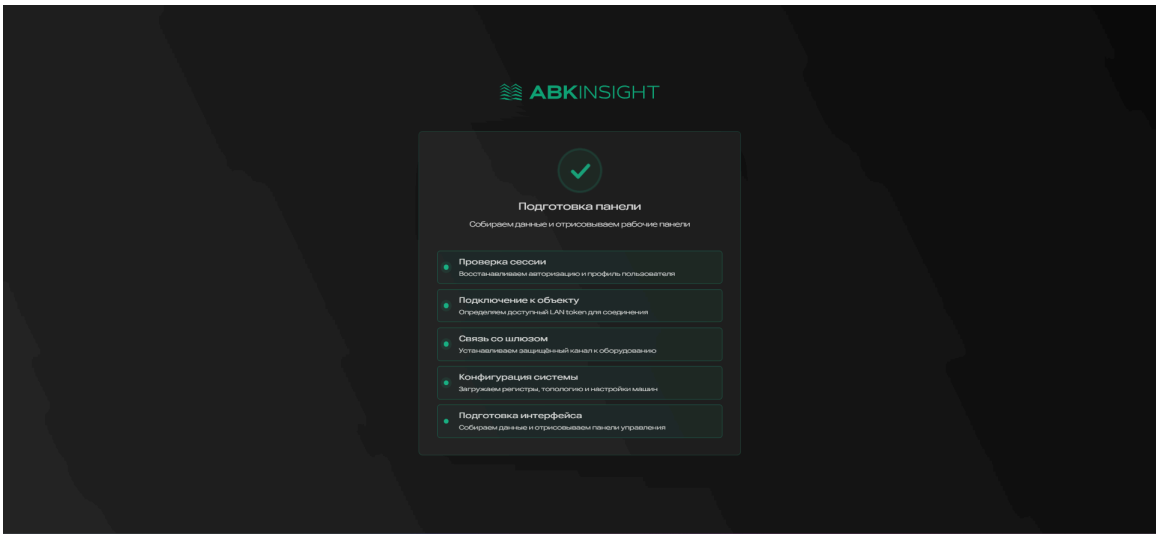
Изменения конфигов отслеживаются LAN-сервером и автоматически доставляются клиентам через `file_update`.

Эксплуатация

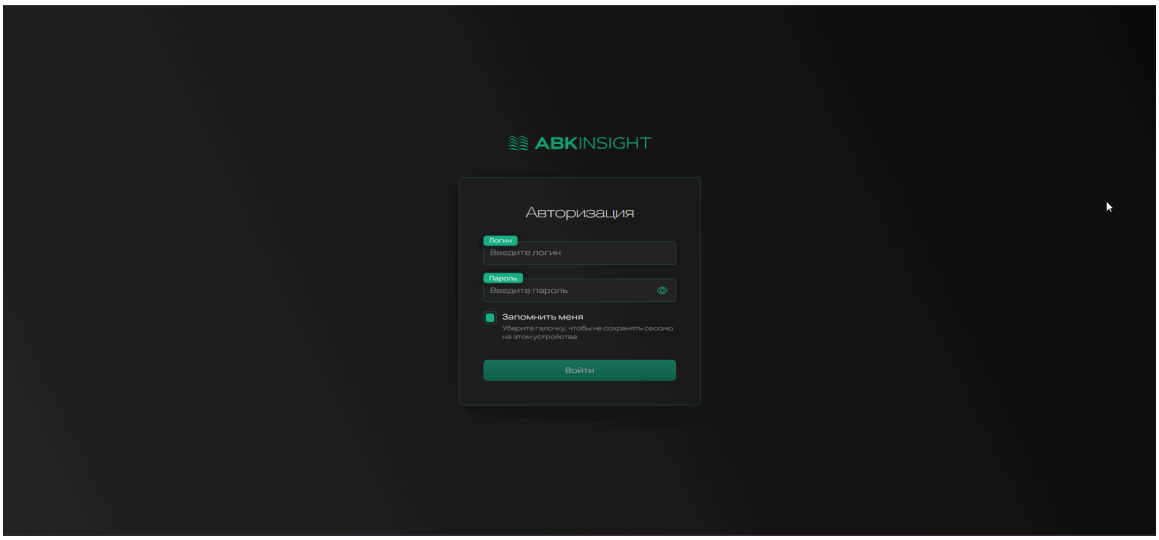
Работа пользователя (web/desktop)

Типовой сценарий:

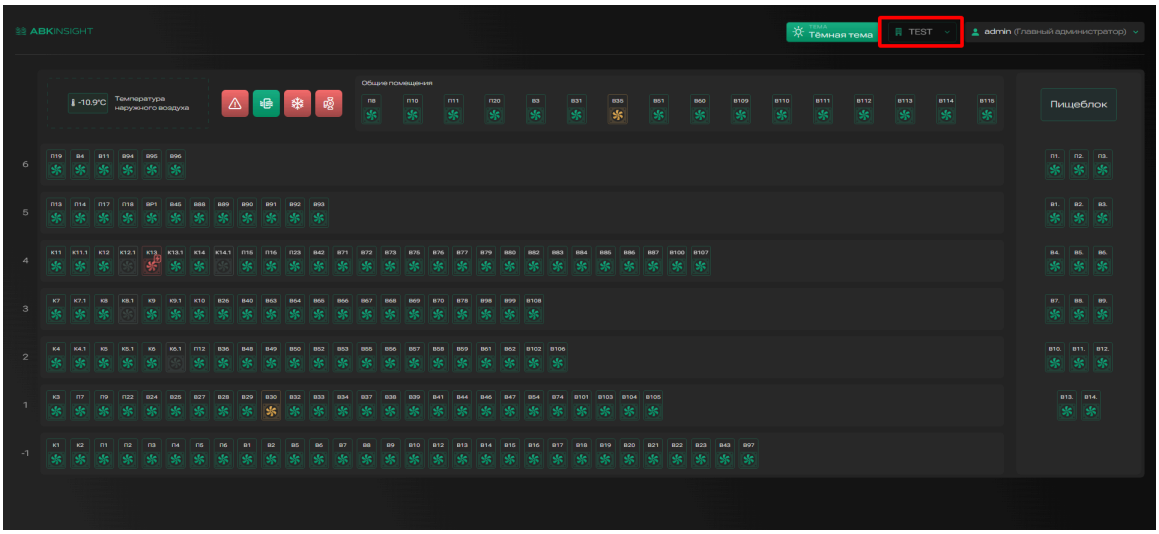
1. Открыть URL интерфейса или файл программы (exe).



2. Авторизоваться (логин/пароль или device-flow для desktop).

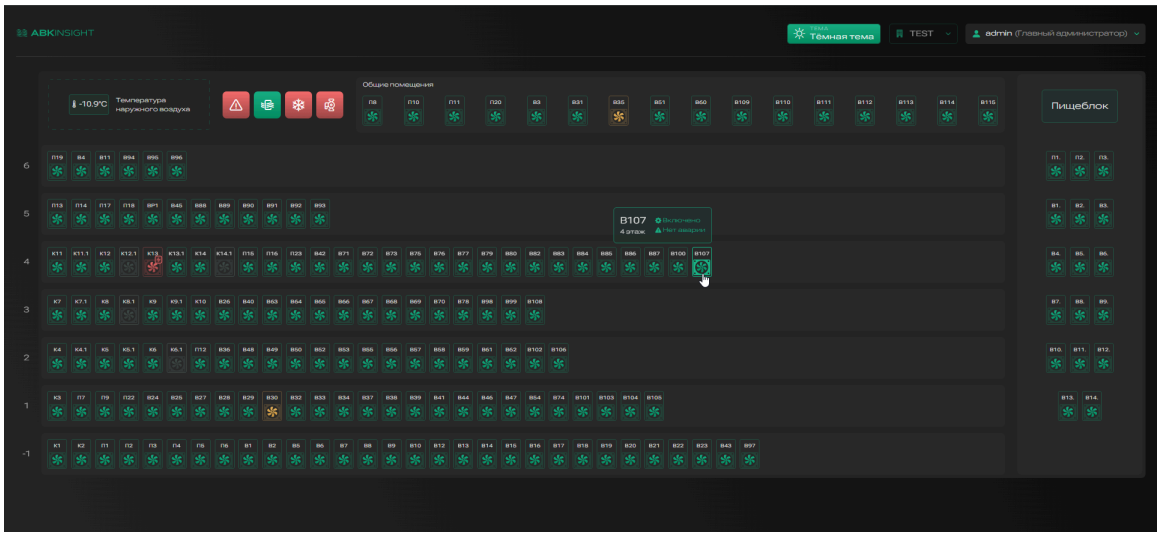


3. Выбрать здание (`lan_token`) в верхней панели.

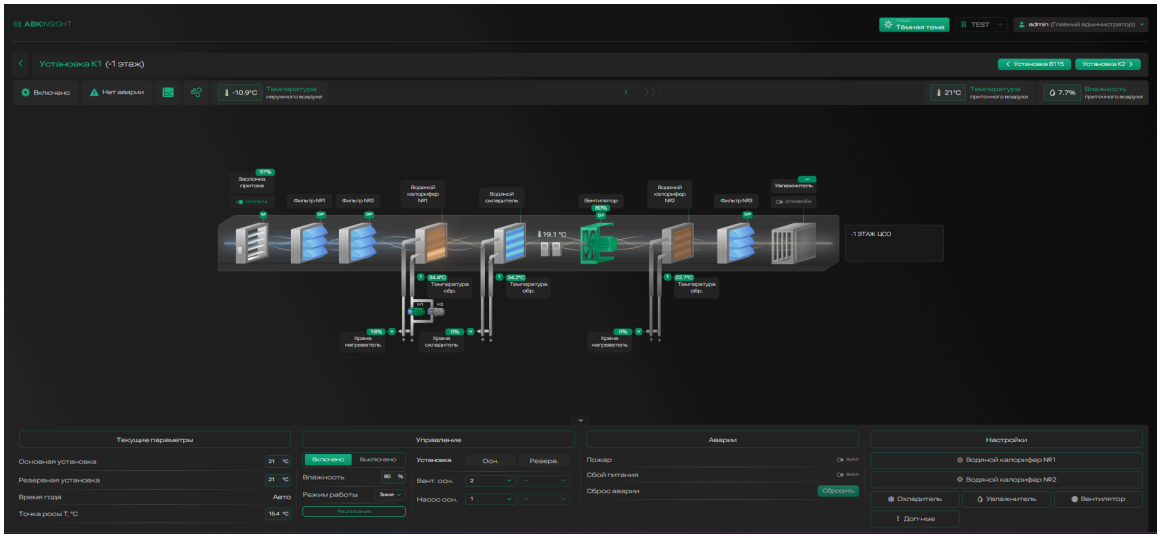


4. Работать с экранами:

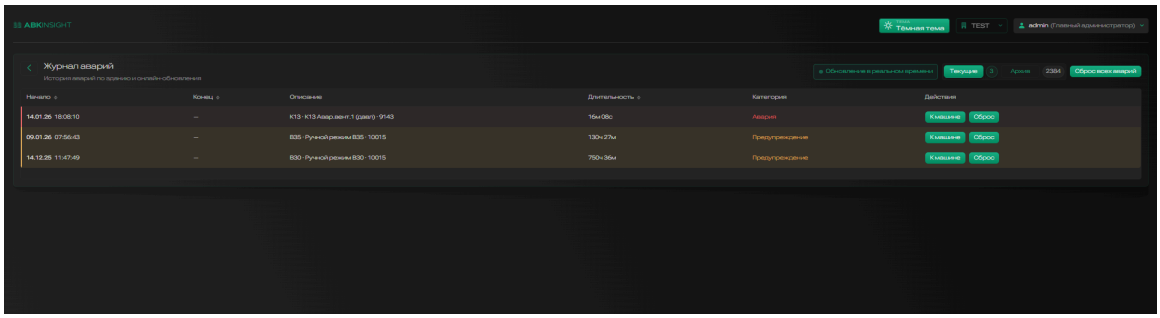
- “Здание” — обзор и быстрый контроль;



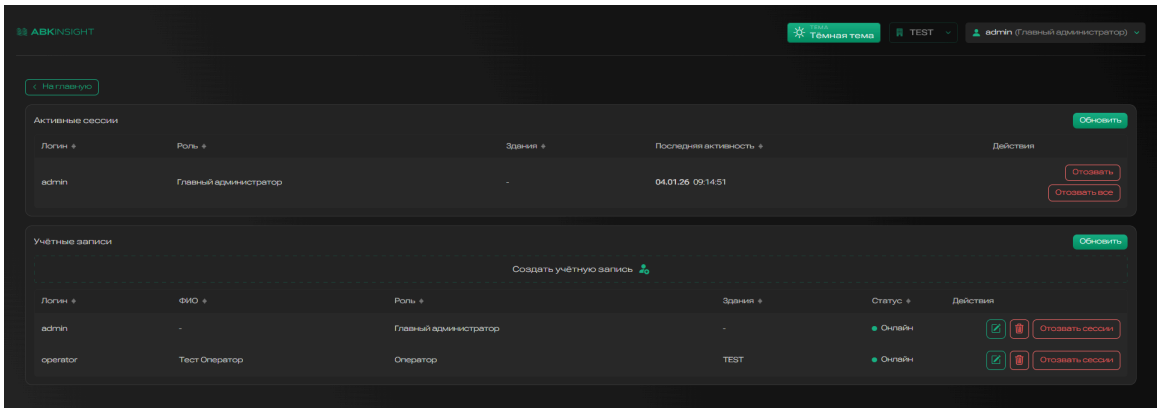
- “Установка” — параметры, уставки и управление;



- “Аварии” — активные/архив, фильтрация и переход к установке;.



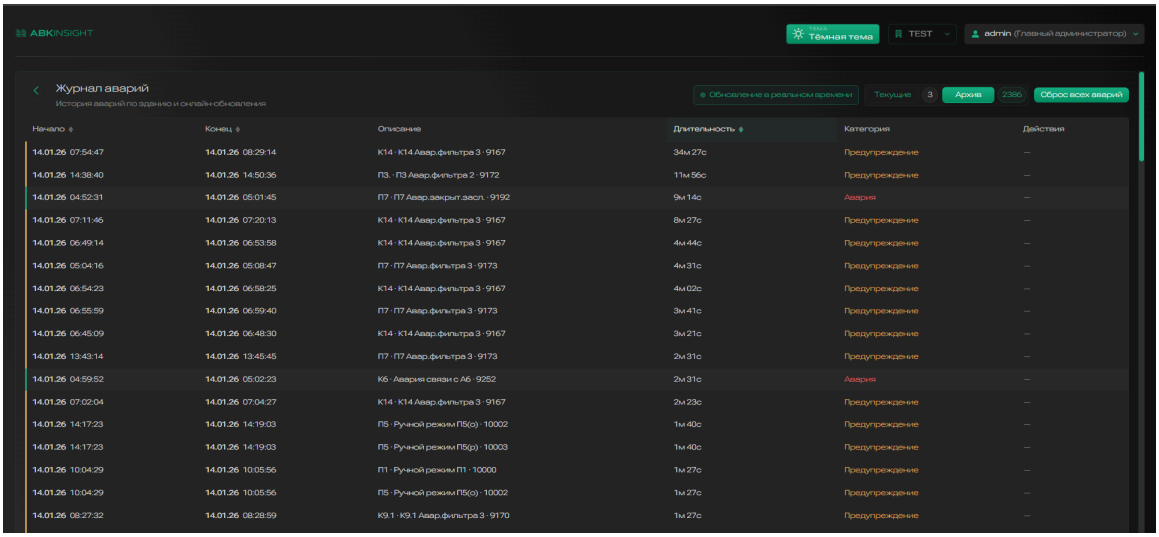
- “Администрирование” — управление пользователями/сессиями (если доступно).



Работа с авариями

Аварии формируются по телеметрии регистров, у которых `type` равен `Alarm` или `Warning`:

- при переходе значения в активное состояние создаётся событие “start”;
- при возврате в неактивное состояние авария закрывается (фиксируется “end” и длительность).



Настройки здания (shutdown_on_fire, outside_temperature_mode)

Настройки хранятся локально в LAN-сервере (SQLite, LAN_DB_PATH). Шлюз проксирует доступ через /api/building-settings/{lan_token}/{key} и рассылает building_setting_update.

Ключи настроек:

- shutdown_on_fire — отключение установок при пожарной аварии (по умолчанию false, роль admin).
- outside_temperature_mode — режим вычисления наружной температуры (min/avg/max, по умолчанию avg, роль admin).
- Настройки установок: room_machines и заметки хранятся в Postgres и доступны по /api/machine-settings/*; изменения рассылаются machine_settings_update (admin/super_admin).
- Расписания работы: хранятся в LAN-сервере и управляются через /api/machine-settings/{lan_token}/{machine_name}/work-schedule; применяются автоматизацией work_schedule.

Резервное копирование и обновления

Резервное копирование

- Postgres: регулярный `pg_dump` или бэкап тома (в Docker-сценарии — volume `pgdata`).
- LAN-сервер: резервное копирование файла SQLite (LAN_DB_PATH) вместе с конфигами, если используются автоматизации/настройки.
- ClickHouse: бэкап по политике эксплуатации (в Docker-сценарии — volume `chdata`).

Рекомендуется согласовать политику хранения телеметрии (retention) и объём диска под неё.

Обновления

- В Docker-пакете `avk-scada-docker` обновление фронта/шлюза происходит автоматически (`git pull` + рестарт приложения).
- LAN-сервер может обновляться централизованно через шлюз (/api/dev/lan/update → /api/lan/update).
- При ручном развертывании — обновляйте код, зависимости и перезапускайте сервисы по стандартной процедуре эксплуатации.

Рекомендации по безопасности

- Размещайте шлюз за TLS-прокси (nginx/traefik) и используйте `wss://` для защиты токенов/данных.
- Ограничивайте доступ к портам шлюза и БД (firewall/ACL), публикуйте только необходимые маршруты.
- Используйте `GATEWAY_WORKER_TOKEN` для защиты `/worker` от посторонних подключений.
- Применяйте rate limiting и лимиты сообщений на уровне прокси, чтобы снизить риск DoS.
- Храните `.env`/токены доступа в защищенном хранилище, ограничивайте права на файлы конфигурации.

Типовые решения (используемое ПО)

В продукте используется ПО сторонних производителей и open-source компоненты, включая:

- Python 3.x (gateway, lan-server, desktop-app);
- FastAPI, uvicorn, uvloop (gateway);
- Postgres (учётные данные, сессии, настройки установок, журнал действий);
- ClickHouse (телеметрия, аварии, графики, статусы);
- SvelteKit + Svelte 5 + Vite + TypeScript (frontend);
- pywebview и PyInstaller (desktop);
- Docker/Compose (опционально для развертывания server-части).

Техническая поддержка

Контакты технической поддержки и регламент обработки обращений определяются договором/регламентом эксплуатации.

Рекомендуемый минимум для заявки:

- описание проблемы и шаги воспроизведения;
- время инцидента и `lan_token`;
- логи `avk-scada-gateway` и `avk-scada-lan-server` (фрагмент с ошибкой);
- при необходимости: выгрузка конфигов registers.json/building.json/outside_temperature_registers.json (с учетом требований безопасности).