SCADA System

User Manual

Version 1.0 — April 2026

Table of contents

Document version: 1.0
Date: April 2026
Interface languages: Multilingual (EN, RU, UK, DE, FR, PL, AR, and others)

Preface

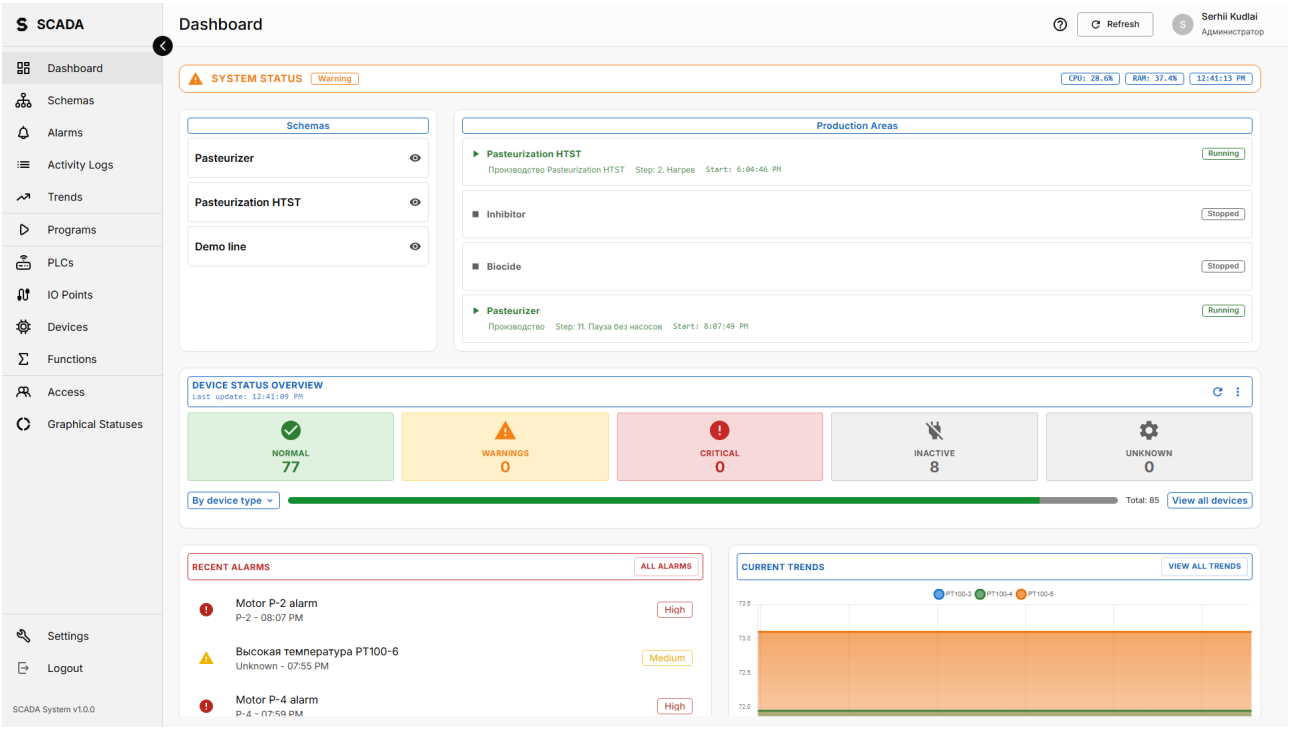
About the system

SCADA System is a modern Supervisory Control and Data Acquisition platform, designed from the ground up for next-generation industrial automation. The system covers the full cycle of process control — from hardware configuration and mnemonic schema design to real-time operation of production lines.

Unlike traditional SCADA systems that date back to the era of desktop applications and proprietary protocols, this platform is built on a modern web stack and microservice principles. This means the operator can work with the system from any browser — on an industrial PC in the shop floor, on a tablet in the lab, or on a laptop in the office — with no additional software to install.

The system covers the full lifecycle of automated production:

- Design — visual mnemonic schema builder, device configuration, creation of control algorithms (sequences) and recipes
- Deployment — automatic PLC code generation (Siemens SCL/XML), I/O point provisioning, deploy via TIA Portal Agent
- Operation — real-time process monitoring, equipment control, trend analysis, batch management
- Maintenance — activity log, alarms, diagnostics, automatic updates, backups



Collage of 4 key system screens: Dashboard (top left), Schema Viewer with a running schema (top right), Trends with charts (bottom left), Sequence Editor (bottom right). Illustrates the breadth of functionality.

Target audience

Role	How they use the system
Process automation engineer	Creates mnemonic schemas, configures devices and sequences, configures the PLC, generates code for TIA Portal
Process technologist	Develops production recipes, configures process parameters, defines alarm thresholds

Role	How they use the system
Line operator	Watches the mnemonic schema in real time, controls equipment (start/stop, setpoints), works with batches and recipes
Shift manager /	Monitors the state of all lines via the Dashboard, analyses trends, works with the alarm journal
System administrator	Manages users and roles, upgrades the system, configures PLC and network connections
Plant management	Reviews reports, efficiency trends, production-line statuses from anywhere via a browser

Key advantages

1. Fully web-based platform

The system runs in any modern browser with no client software to install. This is a fundamental departure from classical SCADA systems (WinCC, Citect, InTouch) that require a thick client on every workstation, are tied to a specific OS (typically Windows) and need separate licenses for each seat.

- Access from any location on the network (intranet or VPN)
- Works on any device: PC, tablet, industrial terminal
- No OS lock-in
- Instant UI rollout to all users simultaneously

2. Real-time data updates

A two-tier WebSocket architecture delivers changes to all connected clients instantly:

- Rust WebSocket server (port 8001) — high-performance delivery of bulk device updates (up to thousands of parameters per second). Written in Rust for minimal latency and maximum throughput.
- Django Channels WebSocket — bidirectional traffic for control commands (motor start/stop, valve open/close, PID setpoint change).

Delta updates transmit only changed fields, while batched sending (every 250 ms) minimises network load when hundreds of parameters change simultaneously.

3. Integrated PLC code generation

A unique capability that has no equivalent in most SCADA systems: from the configuration of devices, sequences, recipes and accidents the system automatically generates a complete project for Siemens TIA Portal:

- 11 function blocks (SCL) for each device type (Motor FB, Valve FB, PID FB, AI/AO/DI/DO FB, COS FB, Counter FB, Timer FB, Tank FB)
- Sequence FB — a state machine for each sequence
- Event / Accident FBs — handlers for events and accidents
- Data Blocks — device instance DBs and recipe DBs
- OB1 Main — main program calling every FB in the right order
- XML Tag Tables — SimaticML tag tables for direct import into TIA Portal

This cuts the time to deploy a new production line from weeks to hours: the engineer configures the process in SCADA, clicks «Generate», uploads the ZIP into TIA Portal — and the PLC is ready to run.

4. Visual mnemonic schema builder

A powerful schema editor built on Konva.js (HTML5 Canvas) removes the need for a separate SCADA graphics editor:

- Drag-and-drop placement of devices, pipes, text and images

- Automatic visualisation of device state (colour, flow animation)
- Support for background images (P&ID drawings as a layer)
- Tank constructor with configurable cap shapes and automatic sensor placement
- Multilingual labels and translations
- Unlimited zoom and panning
- Multi-selection and group operations

5. Flexible recipe and sequence system

The recipe system follows ISA-88 (Batch Control) and cleanly separates:

- Sequence — the algorithm: order of steps and transition conditions
- Step recipe (Recipe) — specific device values for each step
- Production recipe — a set of parameters for a specific product

The same sequence can be used with different recipes to produce different products on the same equipment. Recipe variables parameterise the process without modifying the algorithm.

6. Batch management

A full production batch-management system:

- Creation of batches bound to a recipe and a production line
- Tracking batch progress through line stages
- Queue of batches with priorities
- Automatic transfer between stages
- Full history for each batch

7. AI assistant

An embedded LLM-based AI assistant (via OpenRouter) greatly accelerates work with the system:

- Natural-language control: «Turn on motor MOT_001», «Which devices are in alarm?»
- Automatic P&ID analysis: upload an image → device and connection recognition → placement suggestions on the schema
- Context-sensitive help and explanations
- Agent loop with tools (up to 15 iterations) for complex tasks

8. Multilingual support

Full support for 10+ interface languages, plus the ability to translate user content, not just the UI:

- Interface: EN, UK, RU, DE, FR, PL, AR (with RTL) and more
- Device, sequence, step, recipe and option names — via the name_translations field
- Schema text — via the translations field
- Schema names — via the name_translations field
- On-the-fly language switching without reload

9. Scalable role-based access

A granular permission system lets you precisely control who can do what:

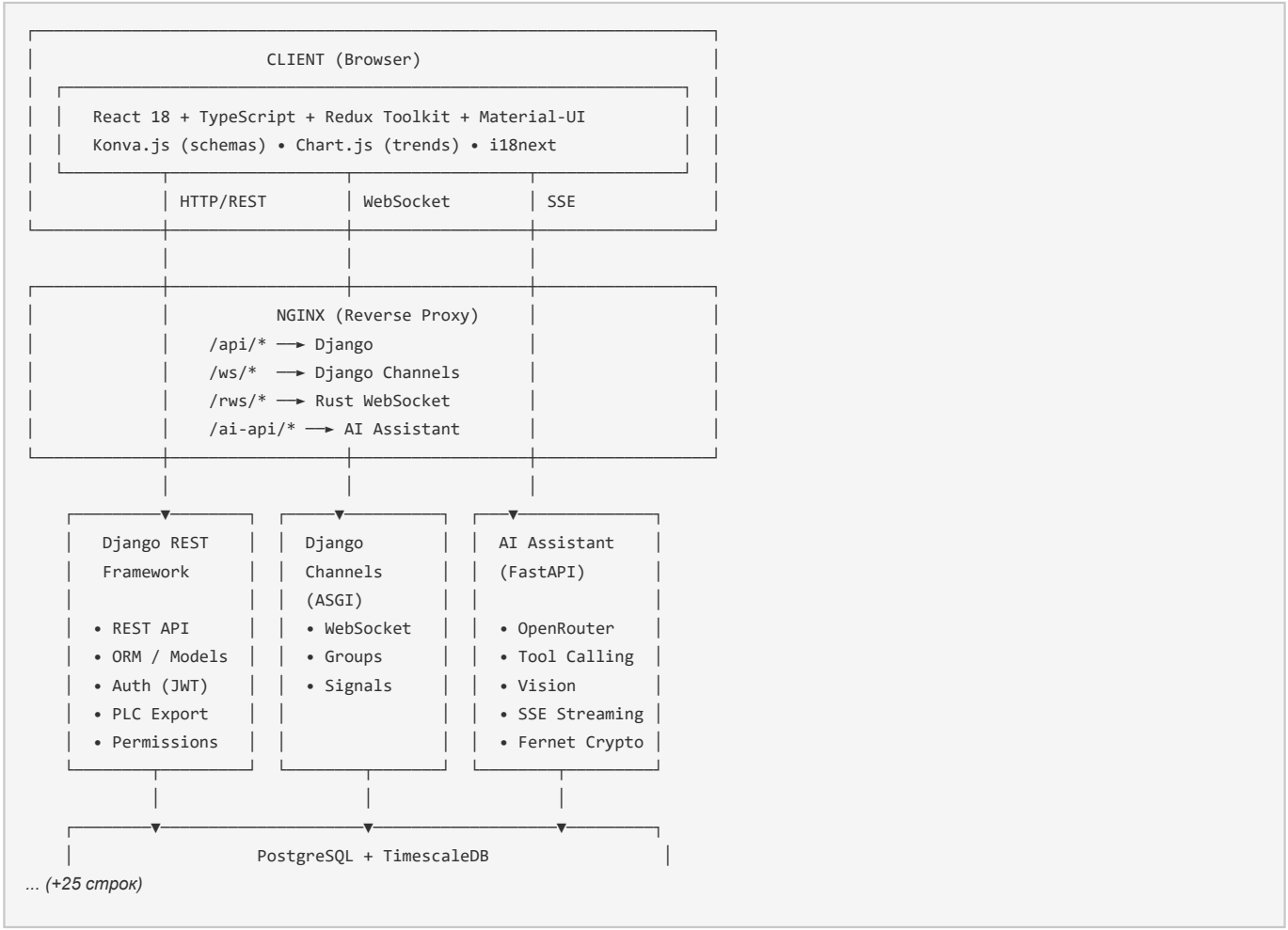
- Roles with a permission matrix across 12 categories and many actions
- Per-schema access control
- Full audit of user actions
- IP filtering for access control

10. Industrial-grade reliability

- Automatic WebSocket reconnection with exponential backoff
- Dual WebSocket: Rust for speed + Django for control (fallback)
- Full DB backup and restore
- Project export/import for migration between installations
- Update system with rollback and dry-run
- Post-upgrade smoke tests

System architecture

The system is built on microservice architecture with clear separation of responsibilities:



... (+25 строк)

Component overview

Frontend (React + TypeScript)

The client application running in the user's browser. Built on React 18 with TypeScript for type safety, Redux Toolkit for state management, and Material-UI for components. Mnemonic schemas are drawn via Konva.js (HTML5 Canvas); trends via Chart.js. Multilingual support is implemented through i18next with 10+ languages, including RTL (Arabic). The frontend talks to the backend via REST API (axios), WebSocket (native API) and SSE (for AI streaming).

NGINX (Reverse Proxy)

The entry point for all requests. It routes traffic between microservices:

- /api/* → Django REST API
- /ws/* → Django Channels (WebSocket)
- /rws/* → Rust WebSocket server (high-performance, read-only)
- /ai-api/* → AI Assistant (FastAPI)
- /media/* → Static files (icons, images)

Django REST Framework (Backend API)

The core of business logic:

- All CRUD operations for 40+ resource types
- Authentication and authorisation (JWT + RBAC)
- User, role and permission management
- PLC code generation (11 function block types, state machine, Data Blocks, Tag Tables)
- Project export/import
- System maintenance (upgrades, backups, diagnostics)

Django Channels (ASGI WebSocket)

Bidirectional real-time communication:

- 17+ WebSocket channel types (devices, schemas, sequences, batches, accidents, math functions, activity, system)
- Redis Channel Layer for group broadcasting
- Handling of control commands (device start/stop, batch management)
- Update signals on Django model changes

AI Assistant (FastAPI)

A standalone AI-assistant microservice:

- FastAPI on Python 3.11 (port 8080)
- Integration with the OpenRouter API (LLMs supporting tool calling)
- Orchestrator agent loop (up to 15 iterations) with 4 tool categories
- P&ID image analysis via a vision model
- SSE response streaming (token, toolcall, toolresult, done)
- Encryption of API keys (Fernet/AES)
- JWT authentication (proxied through Django /api/users/me/)
- Dedicated tables in the shared PostgreSQL DB (prefix ai_assistant_)
- Alembic migrations

PostgreSQL + TimescaleDB

A single database for all services:

- Core tables (core_*) — devices, schemas, sequences, recipes, roles, users
- TimescaleDB hypertables — historical trend data with automatic time partitioning and compression
- AI assistant tables (ai_assistant_*) — sessions, messages, settings

TimescaleDB makes time-series queries orders of magnitude faster than vanilla PostgreSQL at large data volumes.

Redis

The communication bus:

- Channel Layer for Django Channels (WebSocket groups)
- Pub/Sub for notifications between Django and the Rust Worker
- Caching of frequent requests

Rust Worker (scada-worker)

A high-performance real-time processing service written in Rust:

- Sequence Engine (engine.rs) — sequence execution: current-step tracking, transition-condition evaluation, device activation/deactivation, recipe application
- Events Processor — sequence-event processing
- Accident Processor — accident detection, accident-record creation
- Tank Volume Calculator — periodic tank-volume calculation (every 1000 ms) from level and pressure sensor data
- Math Function Calculator — math-function recomputation
- DB Batcher — batched writes of changes to PostgreSQL (I/O optimisation)
- WebSocket Batcher — batched WebSocket updates (250 ms interval)
- Rust WebSocket Server (port 8001) — a dedicated read-only device-status streaming server. Significantly faster than Django Channels for bulk updates.

Rust was chosen for this component because of requirements for minimal latency, predictable memory usage and the ability to handle thousands of updates per second without garbage-collection pauses.

Advantages over traditional SCADA systems

Criterion	Traditional SCADA	This system
Client	Thick client (installed on every PC)	Web browser (nothing to install)
OS	Windows only	Any (Windows, Linux, macOS, Android, iOS)
Licensing	Per-seat (per workstation)	Server license (unlimited clients)
Updates	Manual on each PC	Automatic, centralised (all clients at once)
Mobile access	None or a separate module	Responsive interface out of the box
PLC code	None (manual programming)	Automatic from the SCADA configuration
AI assistant	None	Built in (P&ID analysis, control, help)
Multilingual	Limited (UI only)	Full (UI + content + RTL)
Real time	Proprietary protocols	WebSocket + Rust (sub-second latency)
Recipes	Basic support	ISA-88 (Sequence + Recipe + Production Recipe)
Batch management	Separate module (extra license)	Built in
Trend analysis	Built in, limited	TimescaleDB (petabytes of data, autocompression)
Audit	Basic	Full journal with WebSocket updates
Backup/Restore	External tools	Built in (backup, export, import, rollback)
Deployment cost	High (licenses + installation + configuration)	Low (Docker, automated deployment)
Extensibility	Closed ecosystem	Open REST API + WebSocket + SSE

System requirements

Server side

Component	Minimum	Recommended
CPU	4 cores	8+ cores
RAM	8 GB	16+ GB
Disk	20 GB SSD	100+ GB SSD (for trends)
OS	Linux (Ubuntu 22.04+) or Windows Server 2019+	Linux (Ubuntu 22.04 LTS)

Component	Minimum	Recommended
Docker	Docker Engine 24+	Docker Engine 24+ with Docker Compose v2
PostgreSQ	15+ with TimescaleDB	16+ with TimescaleDB 2.x
Redis	7+	7+

Client side (browser)

Component	Requirement
Browser	Chrome 90+, Firefox 90+, Edge 90+, Safari 15+
Screen resolution	Minimum 1280×720, recommended 1920×1080+
Network	Stable connection to the server (LAN/VPN)

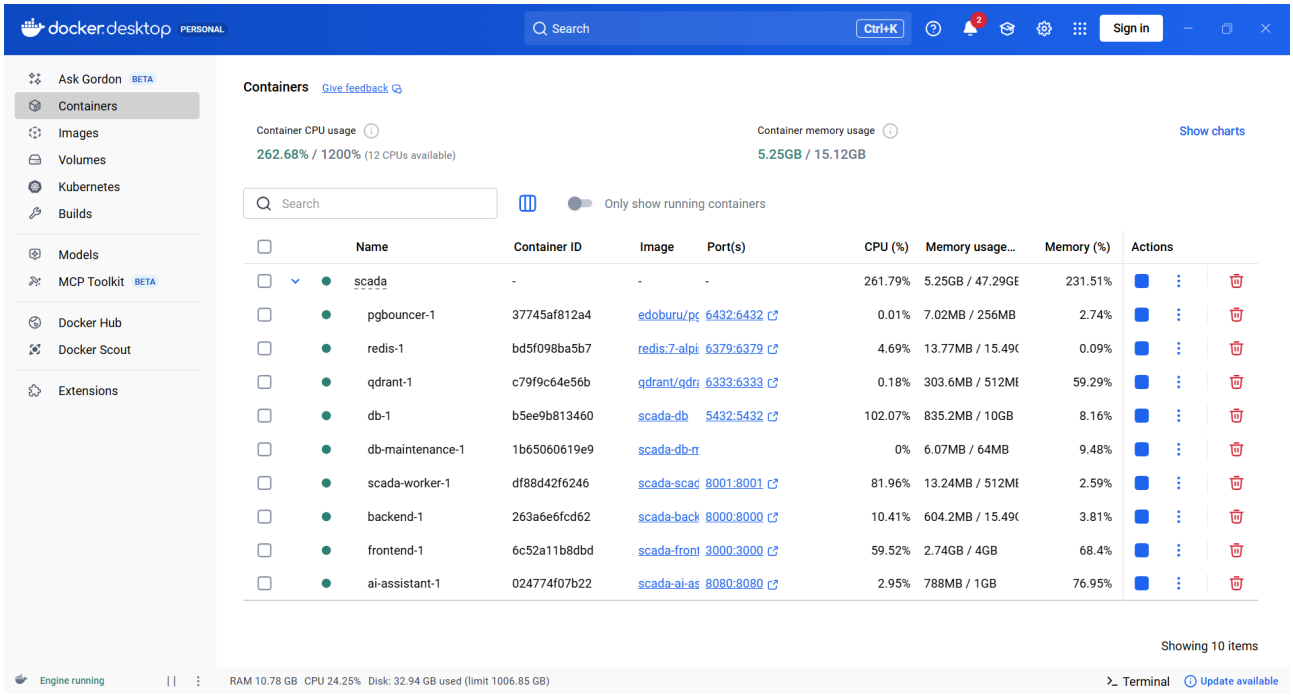
Deployment

The system ships as Docker containers and is deployed with a single command:

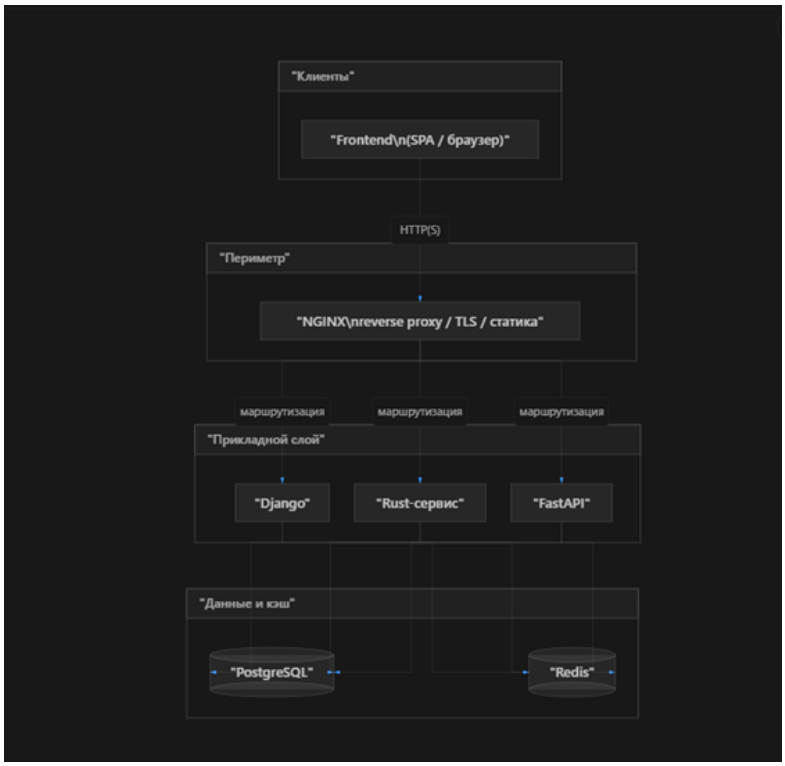
```
docker-compose up -d
```

Container inventory:

- backend — Django REST API + Django Channels
- scada-worker — Rust Worker
- ai-assistant — FastAPI AI service
- postgres — PostgreSQL + TimescaleDB
- redis — Redis
- nginx — Reverse Proxy



Terminal with `docker-compose ps` output: all 6 containers in «Up» status, their ports and uptime.



System architecture diagram: component blocks connected by lines (REST, WebSocket, Redis Pub/Sub, SQL). Colour coding: blue — Frontend, green — Django, orange — Rust, purple — AI, grey — infrastructure.

Quickstart — 15 minutes

This section walks you from a clean installation to a working mnemonic schema with one motor and one valve. If this is the first time you are launching SCADA, start here.

Step 1. First login (2 min)

- Open the installation address in a browser (e.g. http://localhost or http://scada.local).
- On the login screen enter the administrator credentials created by the installer (by default admin with the password set during installation).
- On first login the system may request license activation — enter the key and press Activate. Without an active license most functions are restricted.
- After a successful login you land on the Dashboard.

> Tip. Pick the interface language in the bottom-left corner of the login form — it is saved per user.

Step 2. Create the first PLC (2 min)

Even if you are working only in simulation, the system requires at least one PLC for devices to attach to.

- In the sidebar open PLCs.
- Press «+» in the top right.
- Fill in:
- Name: PLC_DEMO
- IP Address: 192.168.0.100 (can be changed later)
- Type: Siemens S7-1200 (or any available option)
- Rack / Slot: 0 / 1
- Press Save. The PLC status will be «Offline» — that is normal for a demo setup.

Step 3. Create the first devices (3 min)

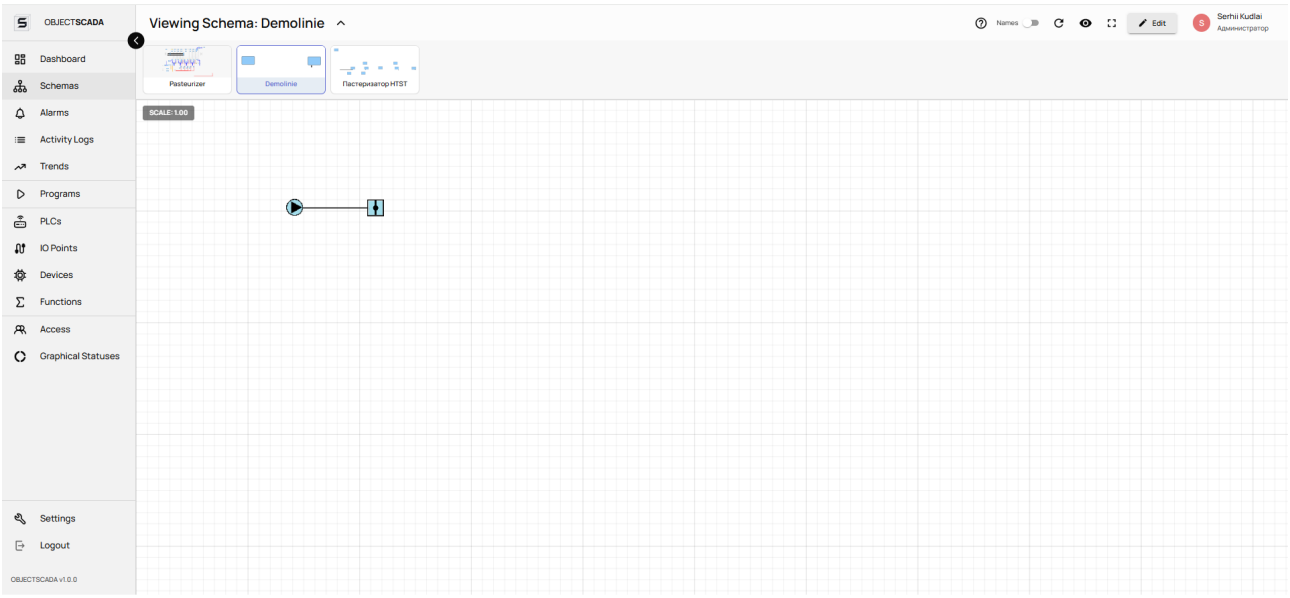
Add one motor and one valve.

- Go to Devices → «+» → Motor.
- Fill in:
- ID: mot_001
- Name: Supply pump
- PLC: PLC_DEMO
- Description: «Demo motor»
- Press Save.
- Similarly create a valve («+» → Valve):
- ID: val_001
- Name: Line valve
- PLC: PLC_DEMO

> Important. The device ID is a text identifier (mot_001, not a number). It is used in every WebSocket channel, in the PLC export and in transition conditions. Pick a naming scheme up front and stick to it.

Step 4. Create the first mnemonic schema (3 min)

- Open Schemas → «+».
- Enter Name: «Demo line» and press Create.
- In the schema list click the pencil (Edit) icon on the new schema — the Schema Builder opens.
- In the left toolbar press Device (); in the dialog pick mot_001 → Add. The motor appears on the canvas — drag it into place.
- Press Device again → val_001 → Add — place the valve next to the motor.
- Press Pipe (), click the motor, then the valve — a connecting line appears.
- Press Ctrl+S or the Save button in the top bar.



Simple demo schema: one motor (green circle) connected by a pipe to a valve (yellow triangle), with the canvas grid in the background.

Step 5. Run the viewer and operate it (3 min)

- In the top toolbar press View — you switch to Schema Viewer.
- Click the motor icon — a control popup opens. Switch the mode to Manual and press Start. The indicator turns green — the motor is «running».
- Click the valve → Open. If process monitoring is enabled on the pipe, a flow animation appears.
- Press Stop on the motor — the animation stops.

Congratulations — you have your first working mnemonic schema with live control.

Step 6. What to do next (2 min)

Now you have a basic skeleton. The next steps depend on your goals:

Goal	Where to go
Automate a process (auto start/stop on conditions)	Section 8 «Sequences» and the «Milk pasteurizer» practical example

Goal	Where to go
Configure recipes for different products	Section 9 «Recipes and variables»
Wire up real PLC I/O	Sections 18 «I/O Points» and 19 «I/O Provisioning»
Configure alarm thresholds	Section 10 «Events and Accidents»
Generate a TIA Portal project	Section 27 «PLC Project Export»
Analyse trends	Section 15 «Trends»

Tip. Every page has a «?» button in the header — it launches an interactive guided tour with element highlights.

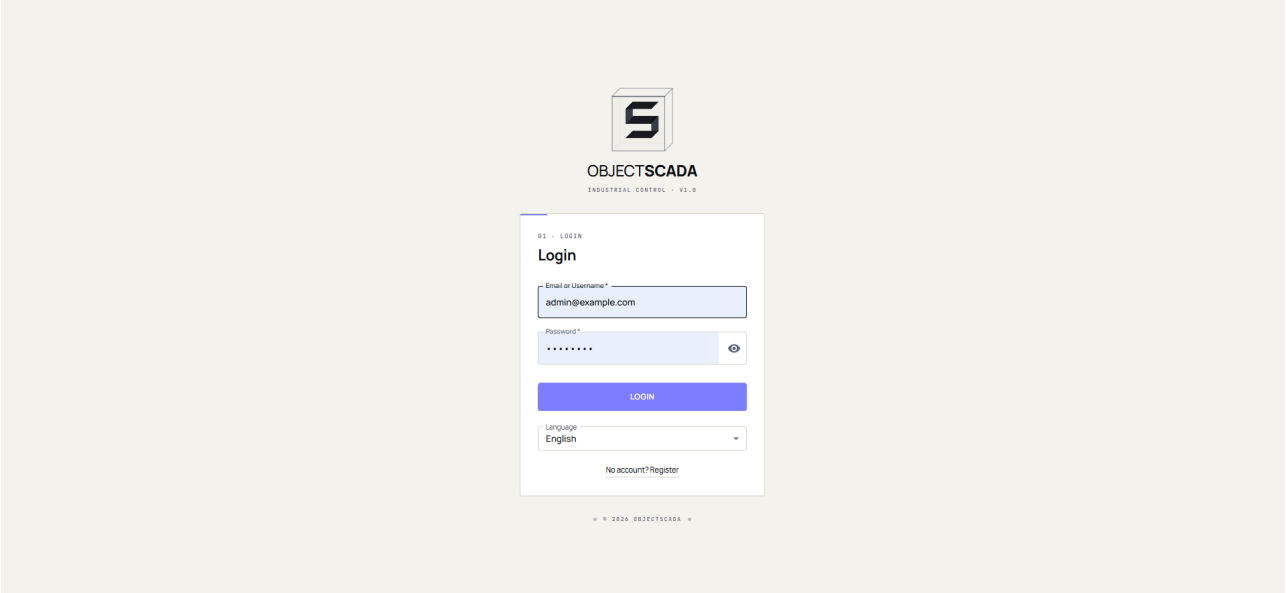
1. Login and registration

For users

The login screen is the first thing you see when the system opens.

Screen elements:

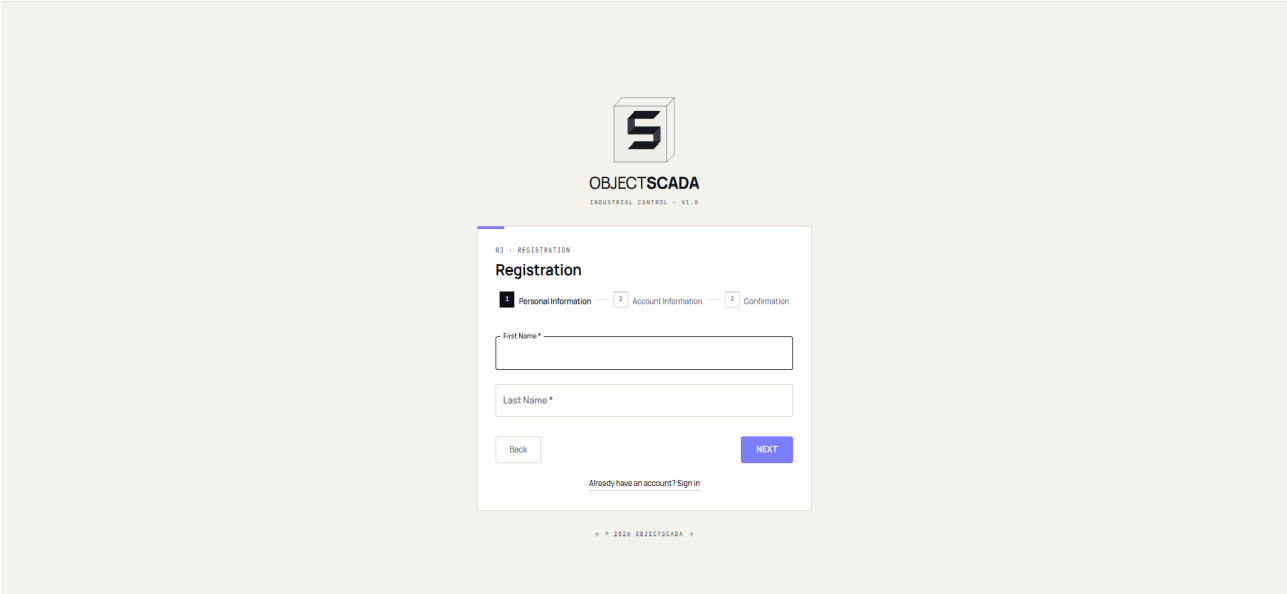
Element	Description
«Username / Email» field	Enter your login or email address
«Password» field	Enter the password. The icon on the right toggles visibility
Language picker	Dropdown to select the interface language (saved after login)
«Sign in» button	Performs the login
«Forgot password?» link	Goes to password recovery
«Sign up» link	Goes to the registration form



Login page: Username/Email and Password fields (with eye icon), language dropdown, Sign in button, Forgot password? and Sign up links. Dark/light theme depending on settings.

Registration screen:

Element	Description
«Username» field	Unique login name
«Email» field	Email address
«Password» field	Password with strength indicator
«Confirm password» field	Password repeat
«Full name» field	First and last name
«Sign up» button	Creates the account



Registration form: Username, Email, Password (with strength indicator), Confirm Password, Full Name, Sign up button.

Technical documentation

Authentication API:

Method	Endpoint	Description	Request body
POST	/api/auth/login	Sign in	{"username": "...", "password": "..."}
POST	/api/auth/register	Register	{"username": "...", "email": "...", "password": "...", "first_name": "...", "last_name": "..."}
GET	/api/auth/me	Current user	—
POST	/api/auth/token/refresh/	Refresh JWT token	{"refresh": "..."}

Login response:

```
{
  "access": "eyJhbGciOi...",
  "refresh": "eyJhbGciOi...",
  "user": {
    "id": 1,
    "username": "admin",
    "email": "admin@example.com",
    "roles": [...]
  }
}
```

Authorization: Every protected endpoint requires the header Authorization: Bearer .

2. Dashboard

For users

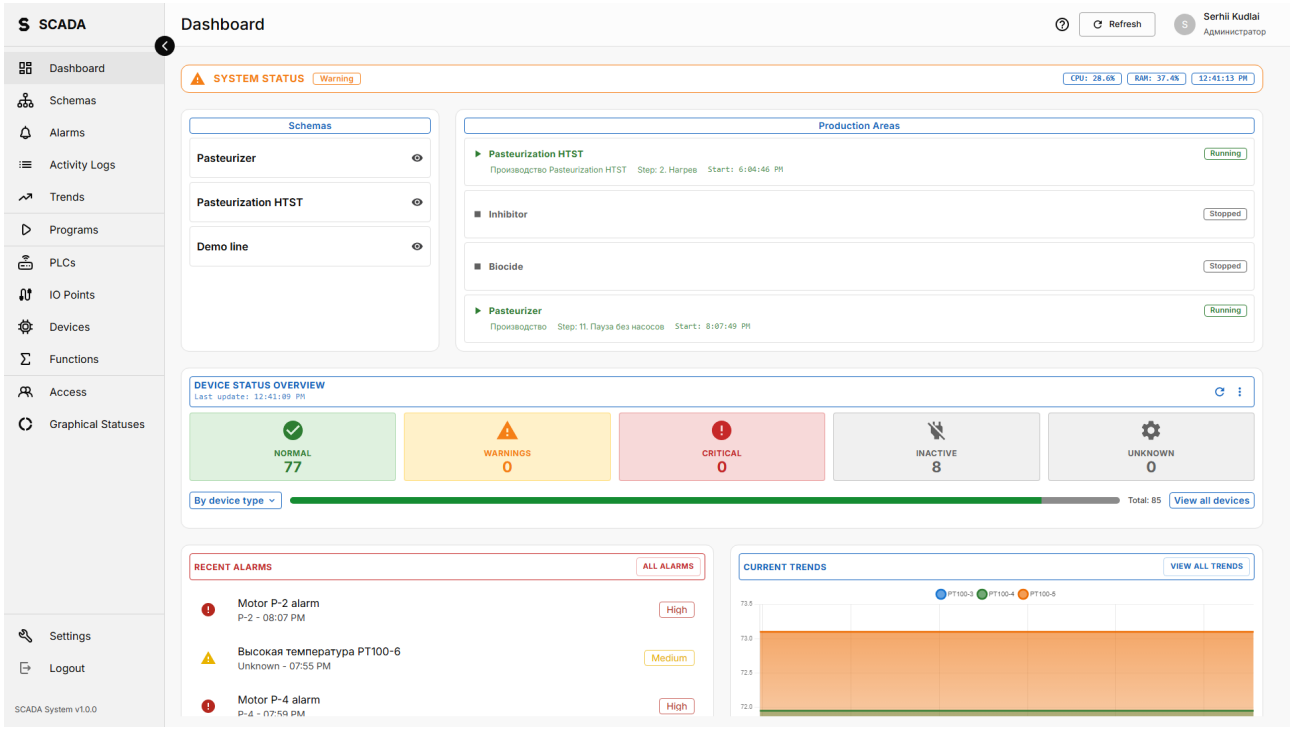
The Dashboard is the central page of the system and shows overall status.

Screen elements:

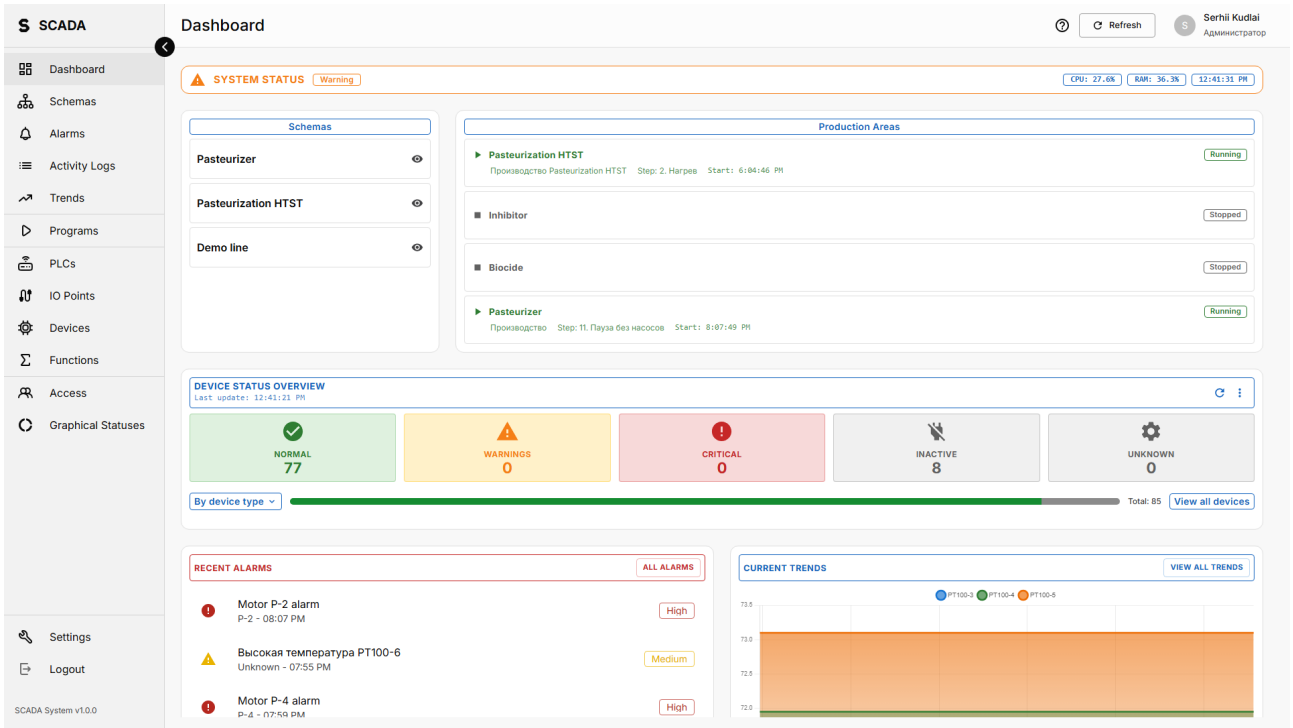
Element	Description
Statistic cards	Quick summary: number of devices, active schemas, accidents, sequences
State overview	Visual indicators of device and system state
Alarm list	Recent alarms with severity colour coding
Mini-trends	Miniature charts of key parameters
Sequence statuses	Current step and progress of every running sequence
System metrics	CPU, RAM, uptime, temperature, disk space
Quick-access buttons	Quick navigation to core functions

Header buttons:

Button	Description
Refresh	Refresh Dashboard data
Tour	Launches the interactive guided tour over Dashboard elements
Export project	Opens the full-project export dialog
Import project	Opens the project import dialog



Full Dashboard view: statistic cards on top, alarm block on the left, mini-trends on the right, sequence statuses below. The navigation sidebar on the left.



System metrics block: CPU usage (%), RAM usage (%), disk usage, uptime, network stats. Each parameter has a colour indicator (green/yellow/red).

Technical documentation

System metrics API:

Method	Endpoint	Description
GET	/api/system/info/	System information
GET	/api/system/health-status/	System health

WebSocket: System metrics (streaming every 2 s)

Connection: ws:///ws/system

```
{
  "type": "system_metrics",
  "cpu": 45.2,
  "memory": 67.8,
  "disk": 34.1,
  "network": {"rx": 1024, "tx": 512},
  "uptime": 86400,
  "temperature": 52.0,
  "lastUpdate": "2026-04-07T12:00:00Z"
}
```

WebSocket: PLC status (streaming every 2 s)

```
{
  "type": "plc_status",
  "is_active": true,
  "status": "running",
  "service_running": true,
  "timestamp": "2026-04-07T12:00:00Z"
}
```

3. Schemas

For users

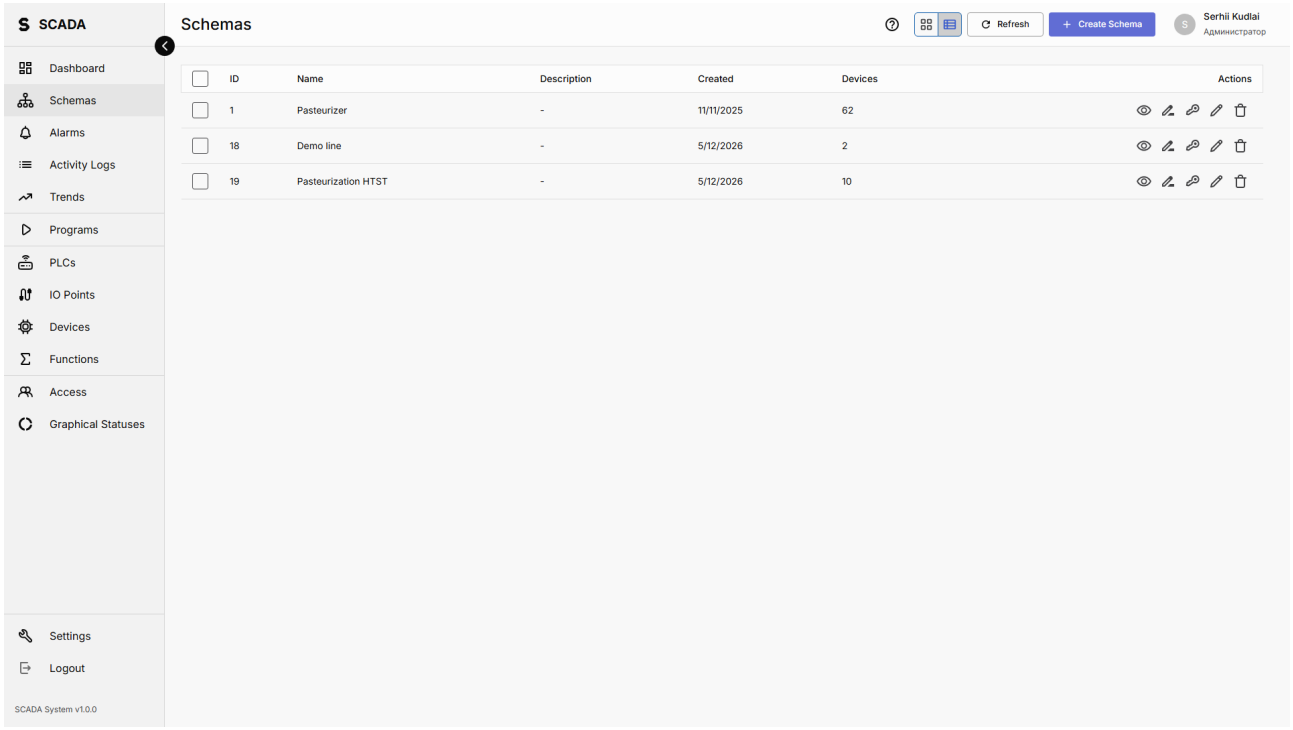
Mnemonic schemas are visual diagrams of your production process. They are the primary tool for monitoring and control.

List screen elements:

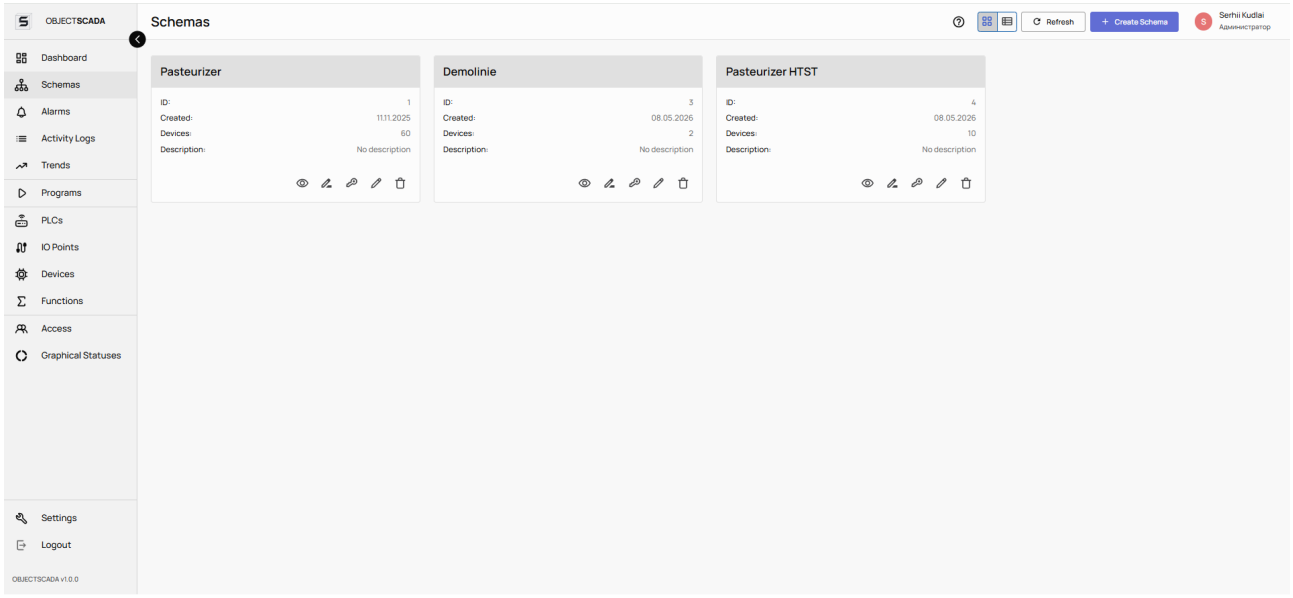
Element	Description
Schema list/grid	Cards or table rows with mnemonic schemas
View toggle	Grid / List (icons in the top right)
«+» button	Create a new schema
Search	Search by name
Refresh button	Reloads the list

Per-schema actions:

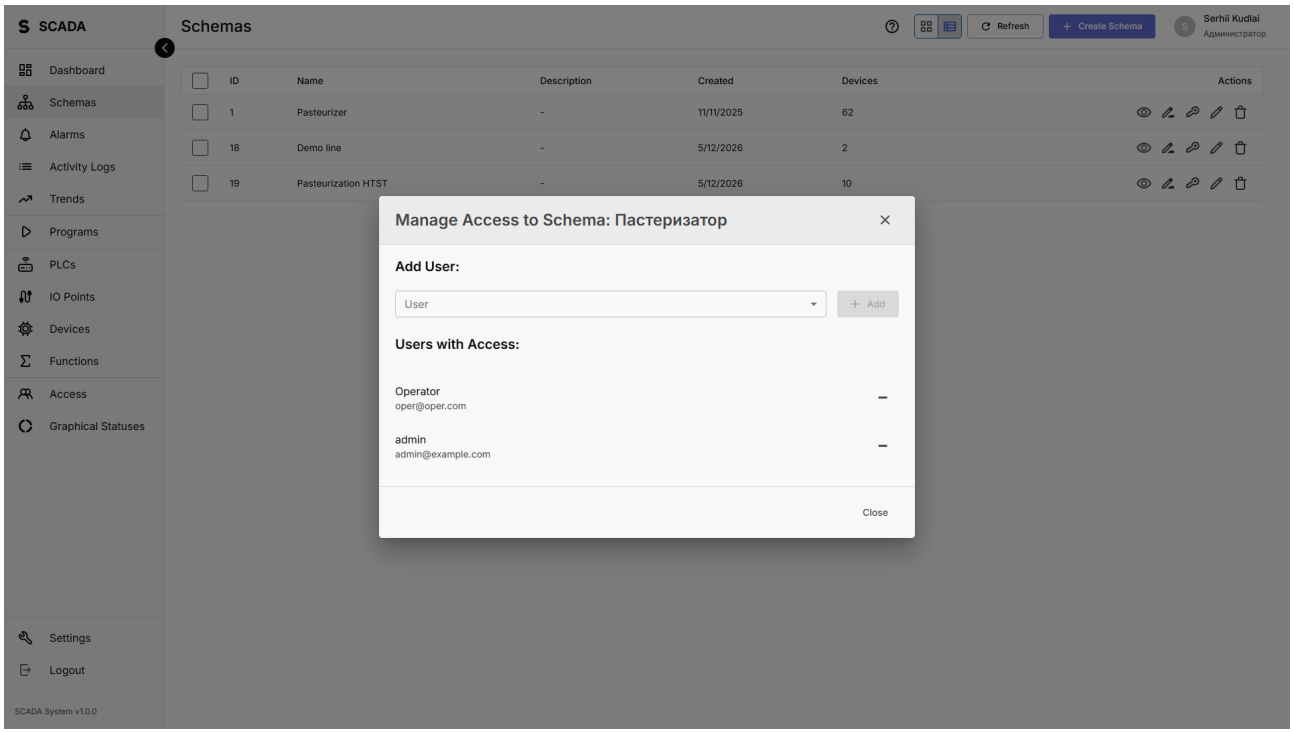
Button	Icon	Description
View		Opens the schema in view mode (Schema Viewer)
Edit		Opens the schema in the editor (Schema Builder)
Delete		Deletes the schema (with confirmation)
Access management		Configures user access to the schema
Rename		Changes the name and name translations
Settings		Schema parameters (description, translations)



Schema list page as a grid of cards. Each card shows a schema preview, name, modified date, action buttons (view, edit, delete). The «+» button in the top right.



The same page in table form: columns «Name», «Description», «Created», «Modified», «Actions».



Access management dialog: list of users with checkboxes, buttons «Add» and «Save».

Technical documentation

Schemas API:

Method	Endpoint	Description	Parameters
GET	/api/schemas/	List of all schemas	?search=...&ordering=...
POST	/api/schemas/	Create a schema	{"name": "...", "description": "...", "name_translations": {...}}
GET	/api/schemas/{id}/	Retrieve a schema	—
PUT	/api/schemas/{id}/	Update a schema	Full schema object
PATCH	/api/schemas/{id}/	Partial update	{"name": "...", "description": "...", "name_translations": {...}}
DELET	/api/schemas/{id}/	Delete a schema	—
GET	/api/schemas/{id}/user_access/	Users with access	—
POST	/api/schemas/{id}/user_access/	Grant access	{"user_id": 1}
DELET E	/api/schemas/{id}/user_access/	Revoke access	{"user_id": 1}

4. Schema Builder

For users

The Schema Builder is a powerful visual tool for creating and editing process diagrams. It works on a canvas with drag, zoom and rotate for elements.

Left toolbar

Button	Icon	Description
Select		Select and move elements
Device		Add a device from the catalogue onto the schema
Pipe/Signal		Draw a pipe (connection between devices)
Text	T	Add a text annotation
Image		Add a background image (P&ID underlay)
Sequence		Bind a sequence to the schema
Option		Add a sequence option button
Recipe variable		Add a recipe-variable display
Production line		Add a production line
Math function	*f*	Add a math function
Tank		Open the tank constructor (separate button)
Create device		Quickly create a new device

Top toolbar

Button	Description
Save	Saves the current state of the schema
Undo (Ctrl+Z)	Undo the last action
Redo (Ctrl+Y)	Redo the undone action
View	Switches to Schema Viewer
«Names» toggle	Show/hide device names on the schema
Schema name	Displays the current schema name

Canvas controls

Button	Description
Zoom in (+)	Increase zoom
Zoom out (−)	Decrease zoom
Grid	Show/hide the canvas grid
Background colour	Change the schema background colour
Background transparency	Background transparency slider
Collapse	Collapse/expand the left toolbar

Right property panel

The panel appears when an element is selected on the schema and depends on the element type:

For devices (DevicePropertyPanel):

Property	Description
Position X, Y	Coordinates on the canvas
Rotation	Element rotation (0–360°)
Size W, H	Width and height
Colour	Element colour (palette)
Mirror	Flip horizontally/vertically
Show name	On/off caption for the device
Graphic status	Bind to a status template
Configuration	Device-specific parameters

For pipes (PipePropertyPanel):

Property	Description
Name	Pipe name
Colour	Line colour
Thickness	Line thickness (px)
Name visibility	Show/hide name
Process monitoring	Enable flow animation

For text (TextPropertyPanel):

Property	Description
Text	Content
Font size	Text size
Colour	Text colour
Bold / Italic	Styling
Translations	Multilingual text translations

For sequences (SequencePropertyPanel):

Property	Description
Position	Coordinates
Size	Width × height
Display	Display settings (name, status, steps)

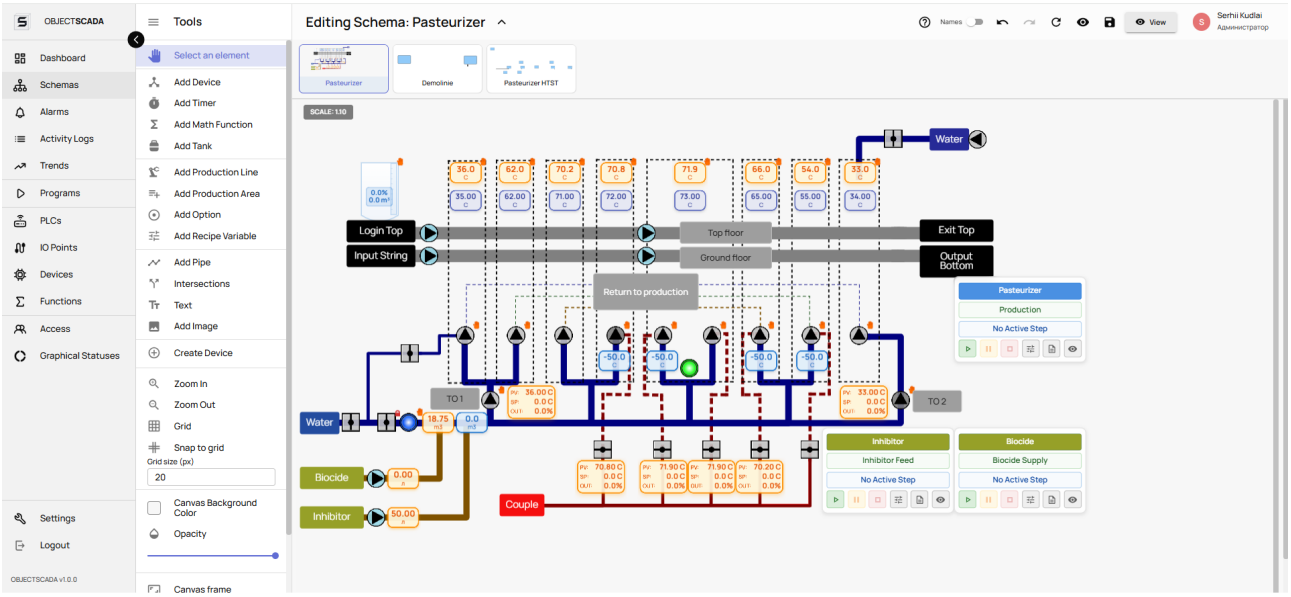
Editor dialogs

Dialog	Description
DeviceDialog	Device catalogue picker for placement on the schema. Filtering by type.
PipeDialog	Pipe name, type and visibility settings
AddSequenceDialo	

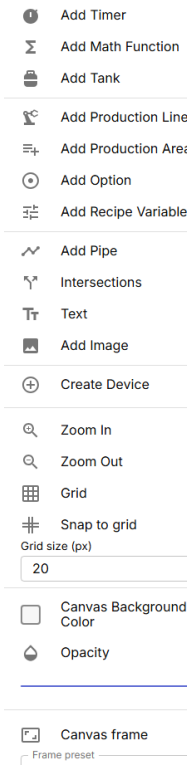
g

Picks a sequence to bind to the schema

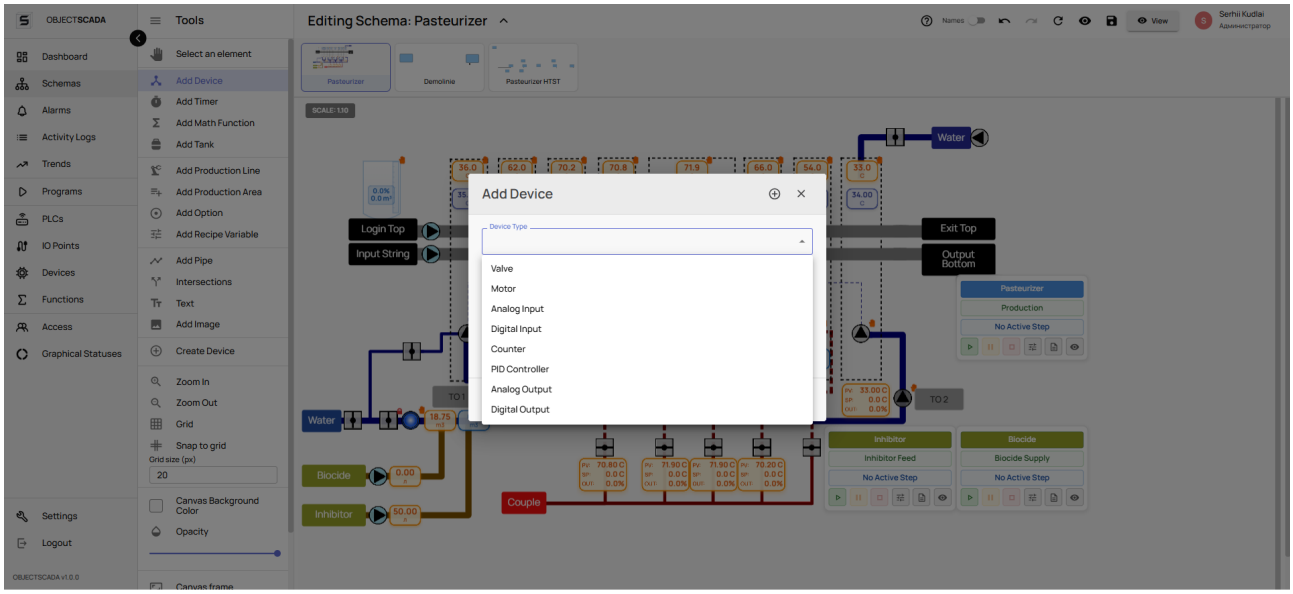
Dialog	Description
AddOptionDialog	Picks a sequence option
AddProductionLine Dialog	Picks a production line
AddRecipeVariable Dialog	Picks a recipe variable
MathFunctionDialog	Picks a math function
TankConstructorDia log	Tank constructor: device selector, cap types (FLAT/DOME/DISH/CONE), dimensions, SVG preview, auto-placement of sensors
ImageDialog	Uploads a background image
TextDialog	Adds/edits text



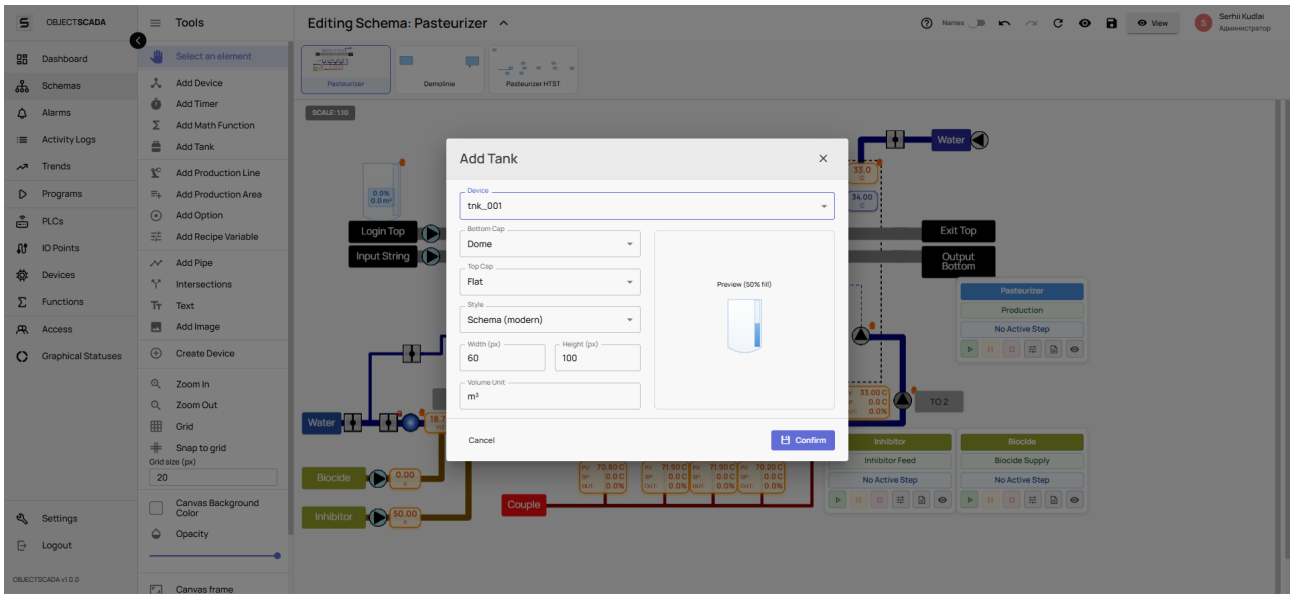
Full editor view: left toolbar with iconic buttons, top bar with «Save», «Undo», «Redo», central canvas with a schema (devices, pipes, texts), right property panel for the selected element.



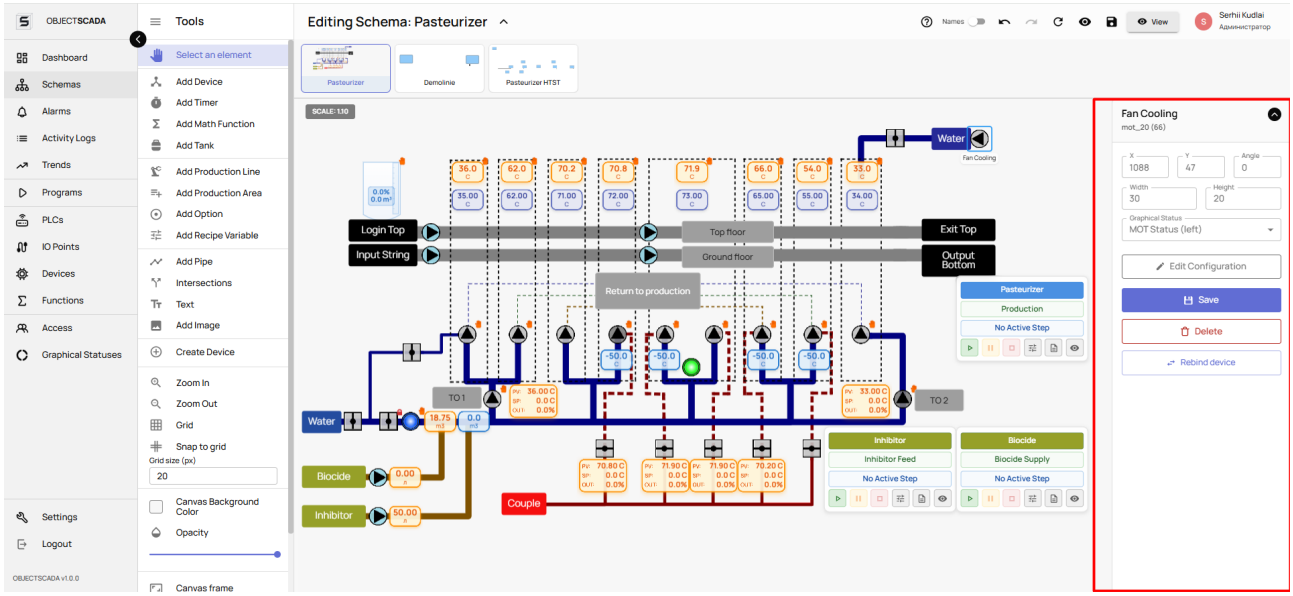
Close-up of the left toolbar with each button labelled (Select, Device, Pipe, Text, Image, Sequence, Option, Recipe Variable, Production Line, Math Function, Tank).



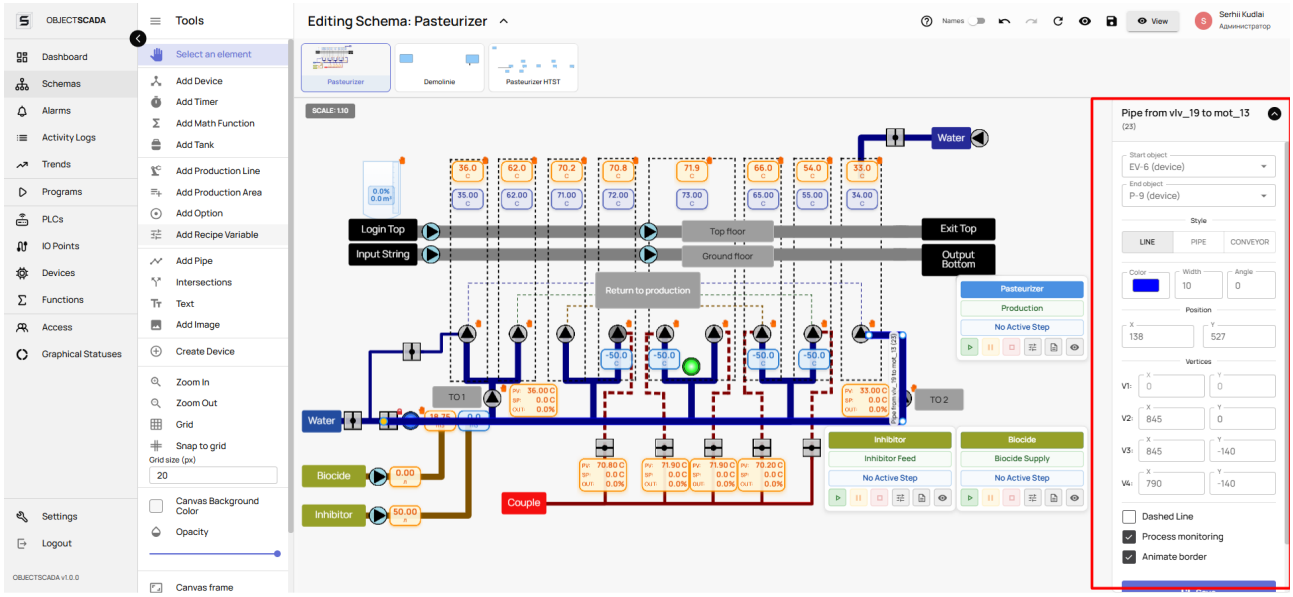
Device selection dialog: list of devices with filtering by type (Motor, Valve, AI, AO, DI, DO, PID, Counter, Timer, Tank, COS), search field, «Add» button.



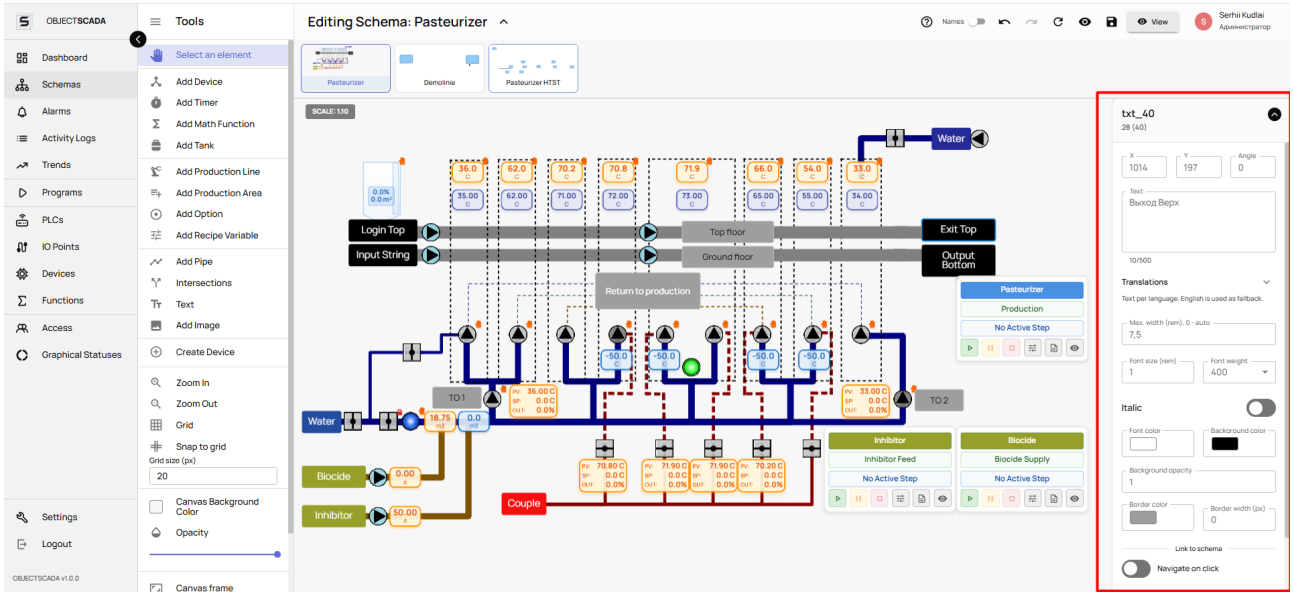
Tank constructor: dropdowns for top and bottom cap (FLAT, DOME, DISH, CONE), width/height fields, SVG preview of the tank shape, tank device selector.



Right property panel when a device is selected: Position X/Y, Angle, Width/Height, Color picker, Mirror/Show Name toggles.



Right property panel when a pipe is selected: Name field, Color picker, Thickness slider, Show Name toggle, Process Monitoring toggle.



Right property panel when text is selected: Text field, Font Size, Color, Bold/Italic toggles, Translations block with fields for each language.

Technical documentation

Schema elements API:

Method	Endpoint	Description
POST	/api/schemas/{id}/add_motor/	Add a motor
POST	/api/schemas/{id}/add_valve/	Add a valve
POST	/api/schemas/{id}/add_analog_input/	Add an AI
POST	/api/schemas/{id}/add_analog_output/	Add an AO
POST	/api/schemas/{id}/add_digital_input/	Add a DI
POST	/api/schemas/{id}/add_digital_output/	Add a DO
POST	/api/schemas/{id}/add_pid_controller/	Add a PID
POST	/api/schemas/{id}/add_counter/	Add a counter

Method	Endpoint	Description
POST	/api/schemas/{id}/add_variable_speed_drive/	Add a VSD
POST	/api/schemas/{id}/add_sequence/	Add a sequence
POST	/api/schemas/{id}/add_option/	Add an option
POST	/api/schemas/{id}/add_production_line/	Add a production line
POST	/api/schemas/{id}/add_recipe_variable/	Add a recipe variable
POST	/api/schemas/{id}/add-math-function/	Add a math function
POST	/api/schemas/{id}/bulk_add_schema_devices/	Bulk add
PATCH	/api/schemas/{id}/update_positions/	Bulk position update
PATCH	/api/schemas/{id}/devices/{deviceId}/	Update an element
DELETE	/api/schemas/{id}/remove_device/	Remove an element

Add-device request body:

```
{
  "device_id": "mot_001",
  "content_type": "motor",
  "position_x": 100,
  "position_y": 200,
  "angle": 0,
  "configuration": {
    "color": "#FF0000",
    "mirror_x": false,
    "mirror_y": false,
    "show_name": true,
    "tank_visual": {
      "bottom_type": "DOME",
      "top_type": "FLAT",
      "width": 60,
      "height": 100
    }
  }
}
```

Pipes API:

Method	Endpoint	Description
GET	/api/pipes/?schema_id={id}	All pipes on the schema
POST	/api/pipes/	Create a pipe
PATCH	/api/pipes/{id}/	Update a pipe
DELETE	/api/pipes/{id}/	Delete a pipe

5. Schema Viewer

For users

View mode offers a live visualisation of the process with real-time data updates.

Features:

Feature	Description
Live device status	Colours and animations update in real time
Click a device	Opens a popup with control and information
Zoom	Mouse wheel or the «+» / «−» buttons
Pan	Right-click drag
Alarm bar	Top/bottom of the screen — current active alarms
Names toggle	Show/hide device captions

Device control popup

Clicking a device opens a tabbed popup:

Control tab:

For each device type:

Type	Controls
Motor	Start/Stop buttons, Auto/Manual toggle, frequency control
Valve	Open/Close buttons, Auto/Manual, %, flow
Analog input (AI)	Auto/Manual, value input, scale indicator
Analog output (AO)	Auto/Manual, value input, Min/Max
Digital input (DI)	Auto/Manual, ON/OFF indicator, interlock
Digital output (DO)	Auto/Manual, ON/OFF buttons, interlock
PID controller	Setpoint, Auto/Manual, P/I/D parameters, error and output
Timer	Start/Stop/Reset, duration setting
Counter	Reset, current value, set value
Tank	Fill level, volume, sensor data

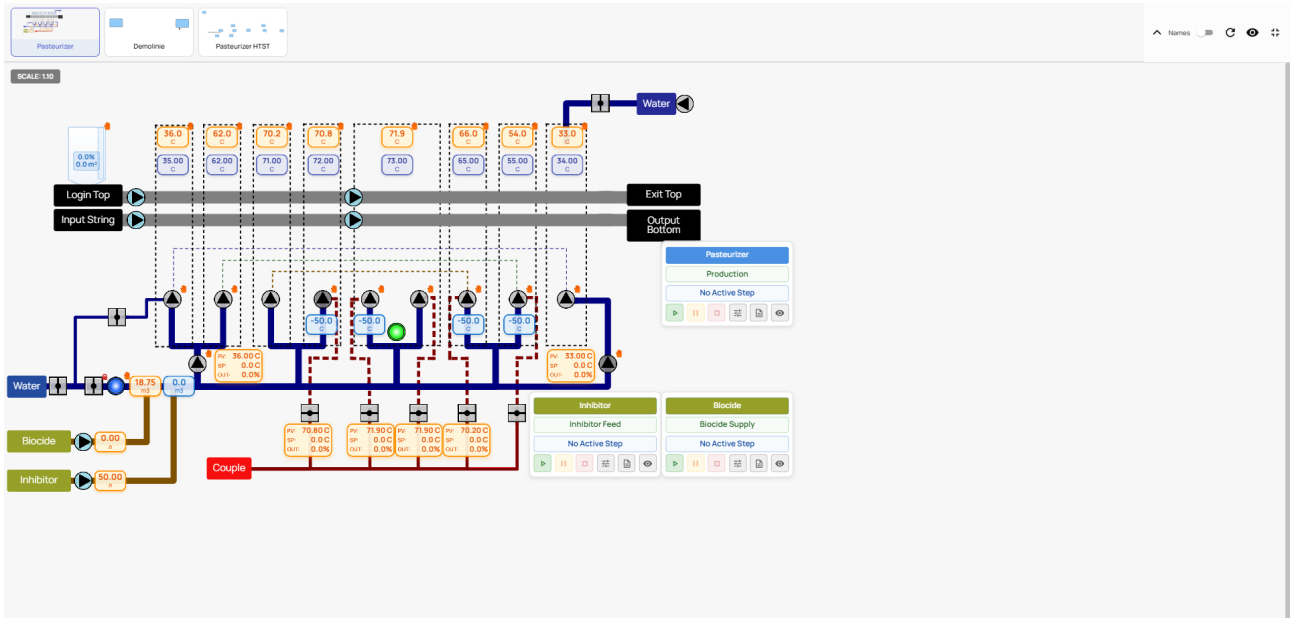
Status tab: Current state in real time.

Table tab: Tabular representation of parameters.

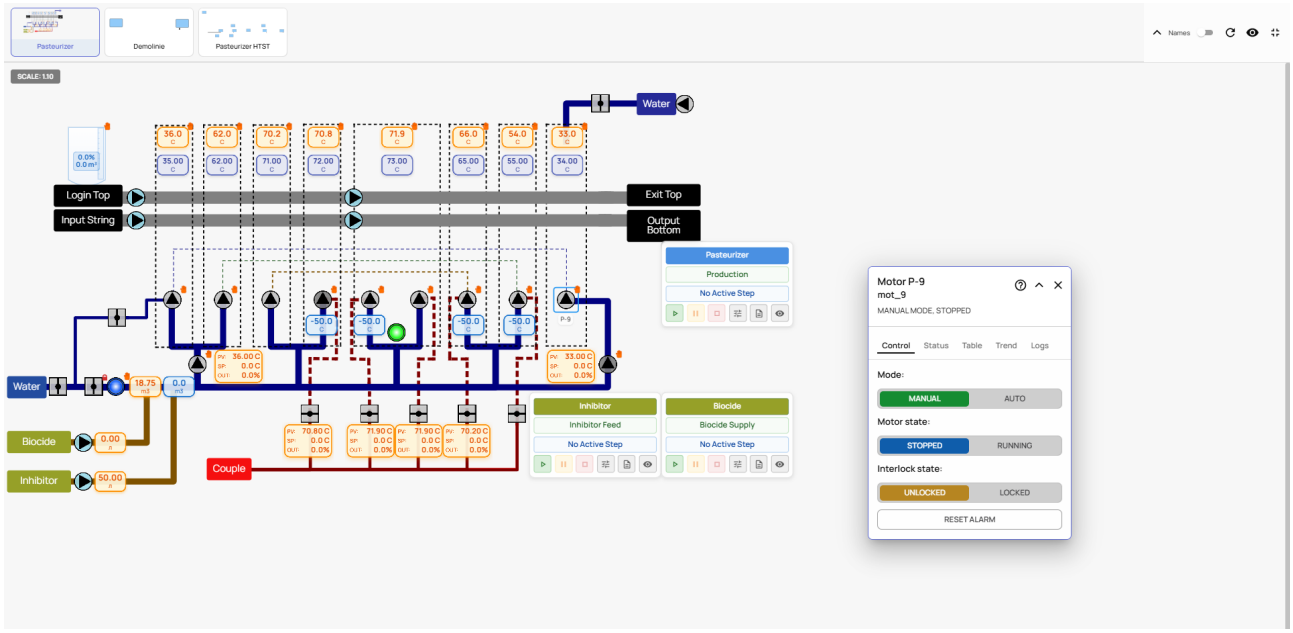
Trend tab: Miniature chart of value history.

Logs tab: Device event log.

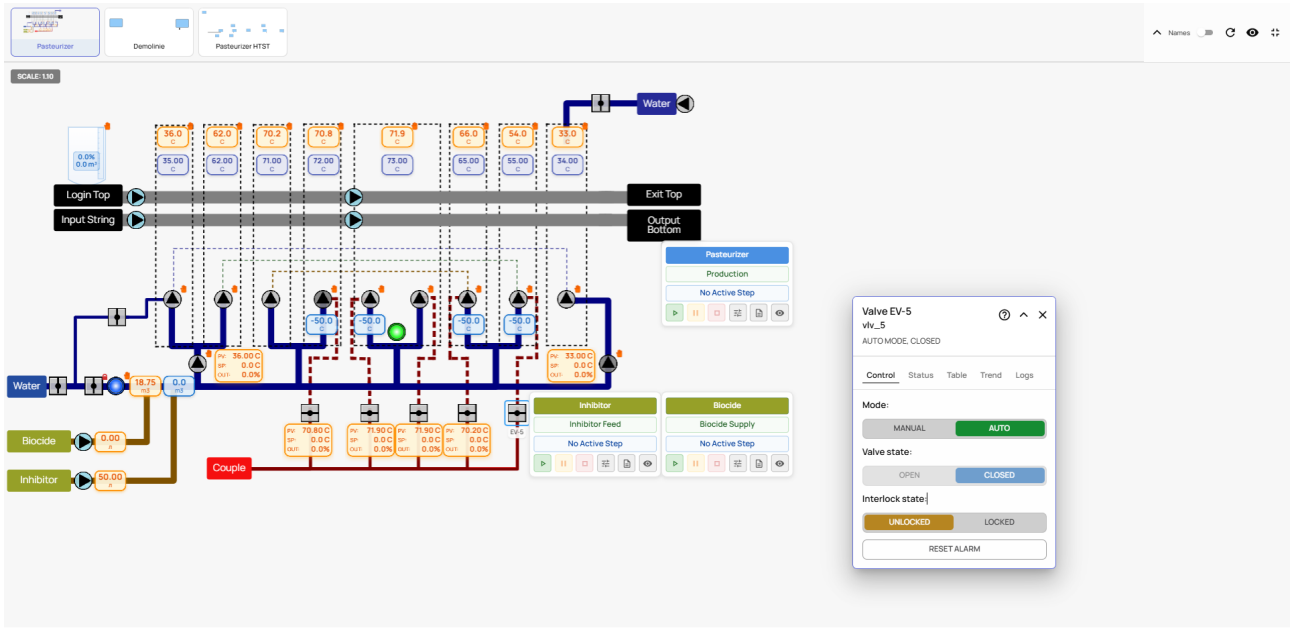
Transition Conditions tab: Active sequence conditions (if the device participates).



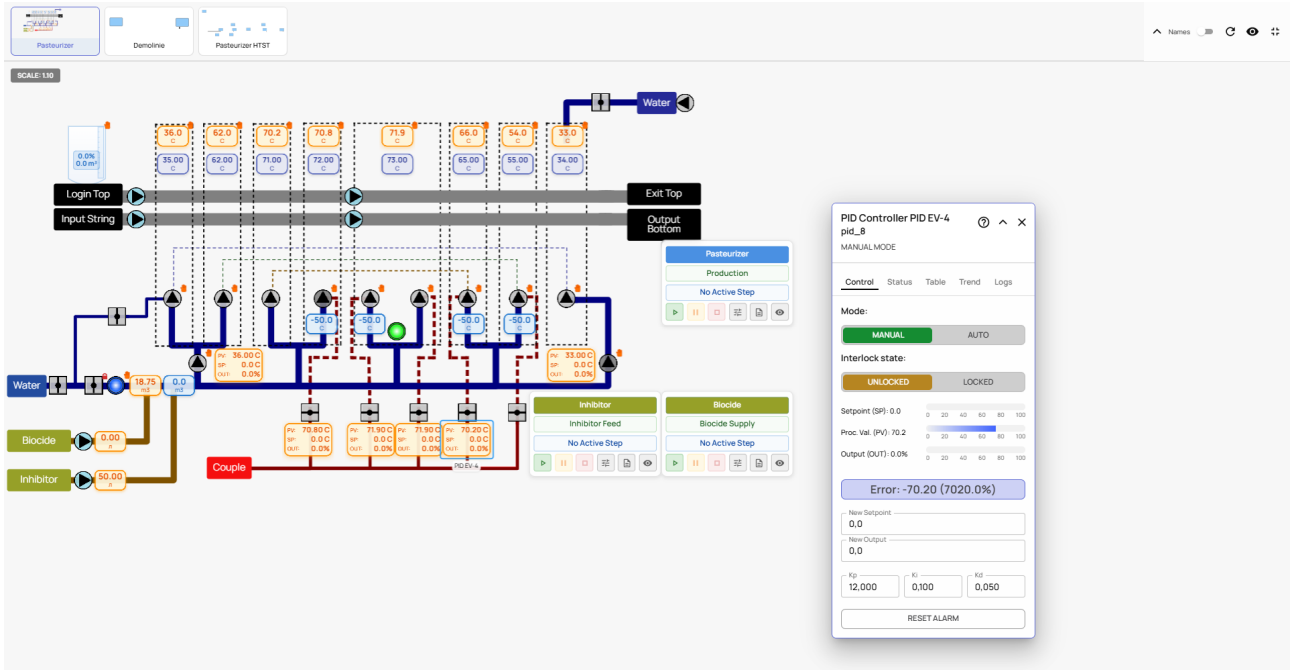
Full Schema Viewer: coloured devices with flow animation along pipes, a sequence block with the current step, and the alarm bar on top.



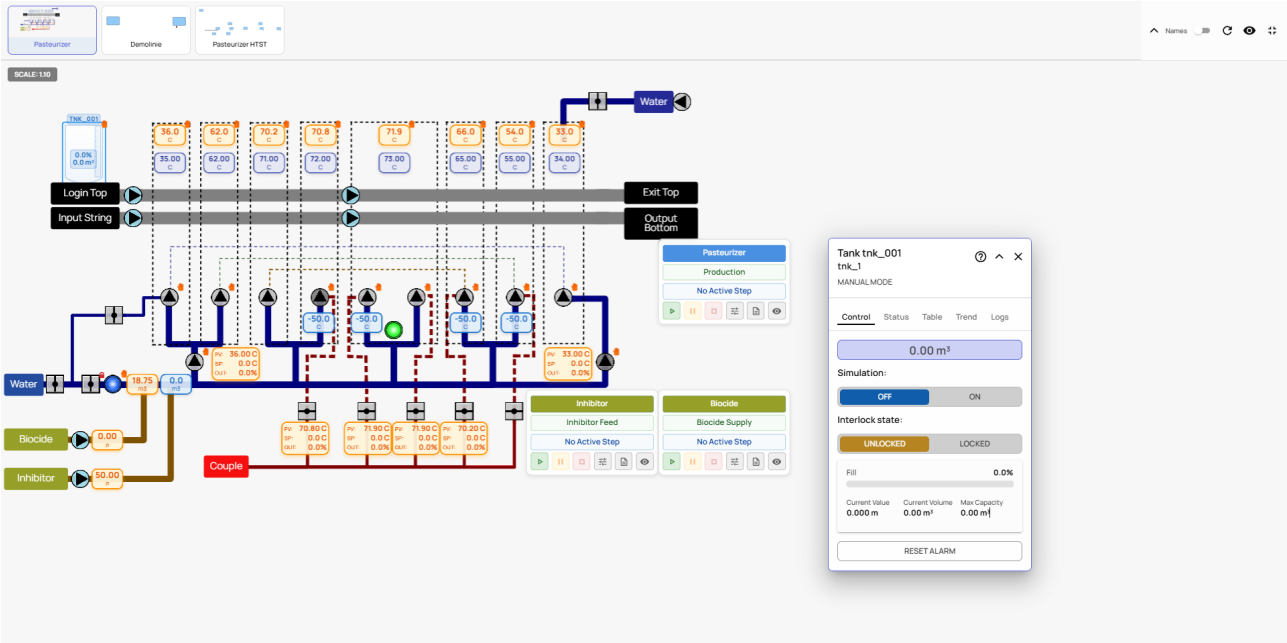
Motor popup: Control tab with Start/Stop buttons, Auto/Manual toggle, frequency. Green indicator — the motor is running.



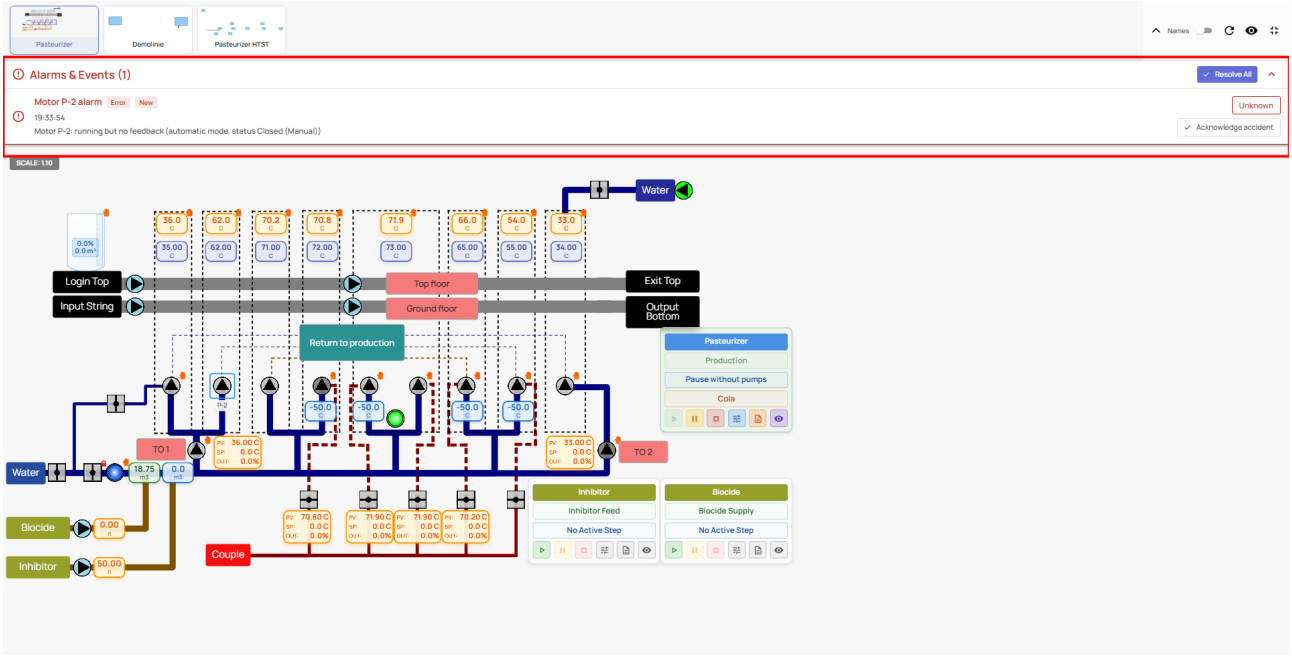
Valve popup: Open/Close buttons, open-percent slider, Auto/Manual mode.



PID popup: Setpoint, P, I, D fields, Error and Output indicators, Auto/Manual toggle.



Tank popup: fill level visualisation, sensor table (level, pressure), volume.



Alarm bar at the top: red/yellow messages with timestamp, severity and device name.

Technical documentation

WebSocket connections for schema viewing:

WebSocket path	Description
ws/schema/{schemaId}/devices	All devices on the schema (real time)
ws/schema/{schemaId}/pipes	All pipes (flow animation)
ws/schema/{schemaId}/sequences	All sequences on the schema
ws/schema/{schemaId}/batches	Batches and production lines
ws/schema/{schemaId}/accidents	Accidents for this schema

Device update event format:

```
{
  "type": "schema_devices_status",
  "devices": {
    "mot_001": {
      "device": "mot_001",
      "current_status": {
        "value": 1,
        "is_running": true,
        "mode": "auto",
        "frequency": 50.0,
        "interlock": false
      },
      "status_icon": "/media/icons/motor_running.svg",
      "position_x": 100,
      "position_y": 200,
      "angle": 0,
      "updated_at": "2026-04-07T12:00:00Z"
    }
  }
}
```

Delta updates (changed fields only):

```
{
  "type": "device_status_update",
  "device": "mot_001",
  "delta": true,
  "changes": {
    "frequency": 45.0
  }
}
```

Device control API:

Method	Endpoint	Description	Body
POST	/api/{device-type}/{id}/set_mode/	Auto/Manual	{"mode": "manual"}
POST	/api/{device-type}/{id}/set_manual_value/	Set value	{"value": 50.0}
POST	/api/{device-type}/{id}/set_simulation/	Simulation	{"simulation": true}
POST	/api/{device-type}/{id}/set_interlock/	Interlock	{"interlock": true}
POST	/api/{device-type}/{id}/update_status/	Update status	{"current_status": {...}}

WebSocket commands (via the frontend websocketService):

```
{
  "type": "device_command",
  "device": "mot_001",
  "command": "start",
  "params": {},
  "options": {}
}
```

Response:

```
{
  "type": "command_response",
  "device": "mot_001",
  "command": "start",
  "status": "success",
  "data": {...},
  "timestamp": "2026-04-07T12:00:00Z"
}
```

6. Devices

For users

The Devices page lets you create, edit and delete all device types.

Device types:

Type	Code	Description
Valve	VLV	Shut-off/control valve
Motor	MOT	Electric motors
Analog input	AI	Analog sensors (temperature, pressure, flow)
Analog output	AO	Analog outputs (4–20 mA, 0–10 V)
Digital input	DI	On/off signals (limit switches, ready signals)
Digital output	DO	Discrete commands
Coordination signal	COS	Inter-system coordination signals
PID controller	PID	Automatic control loops
Counter	CNT	Pulse counters
Tank	TNK	Vessels with volume calculation
Timer	TMR	Software timers

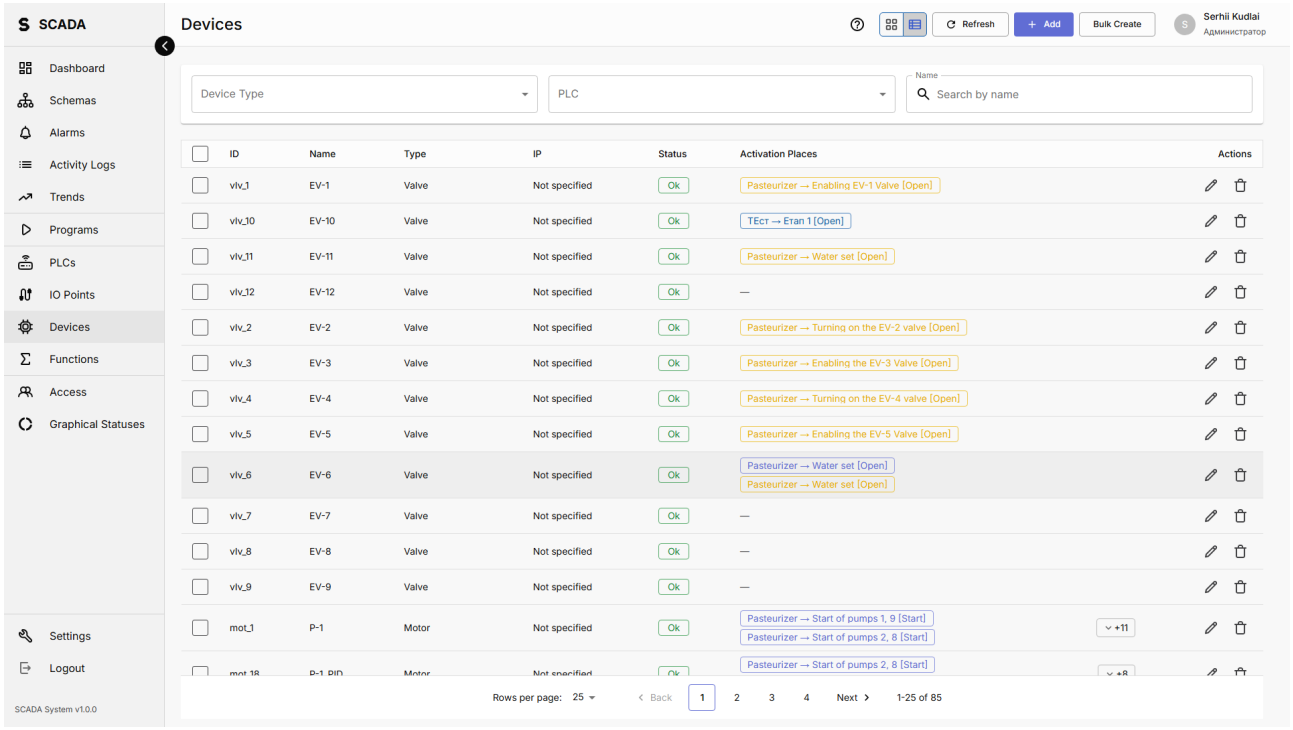
Screen elements:

Element	Description
View toggle	Grid / Table
«+» button	Create a new device
Type filter	Dropdown of device types
Search	Search by name
Refresh	Reload the list

Device form (create/edit):

Field	Description
Device ID	Unique identifier (e.g. mot_001)
Name	Human-readable name
Name translations	Multilingual names
Type	Device type selector
Description	Text description
Default value	Initial value
Engineering unit	EU label
Min / Max	Allowed range
Scaling	Signal conversion parameters
«Save» button	Saves the device
«Cancel» button	Returns without saving

Field	Description
«Delete» button	Deletes the device (edit mode only)



Devices page as a grid: device-type icon cards (motor, valve, etc.) with name, type and value. Type filter on top.

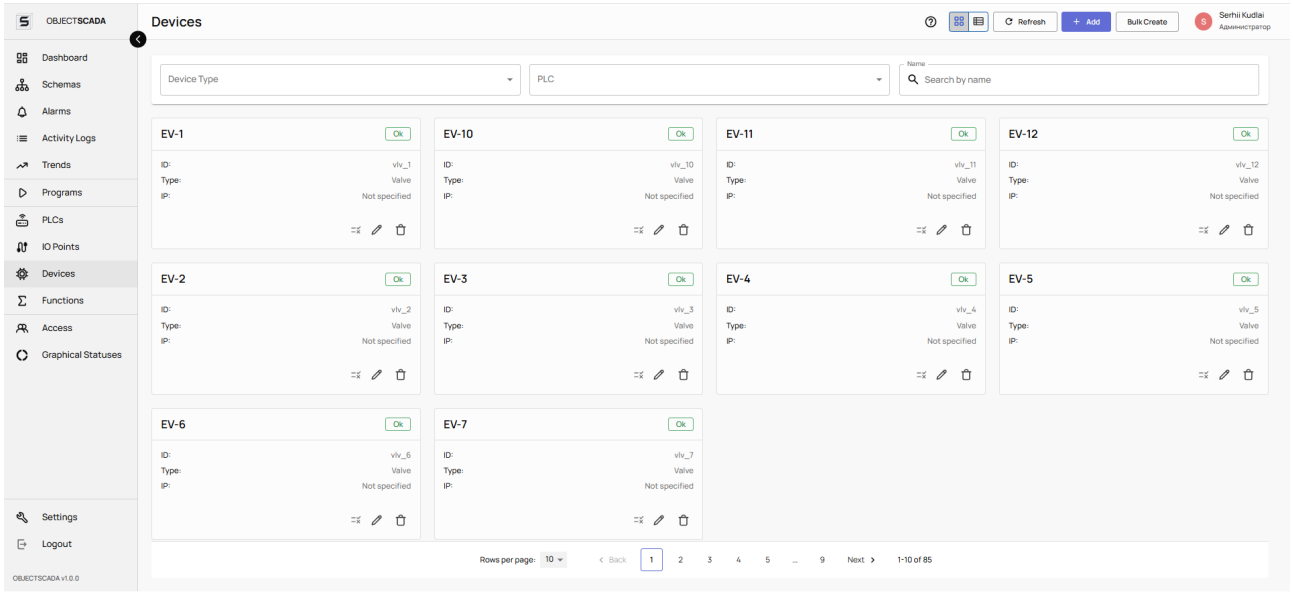
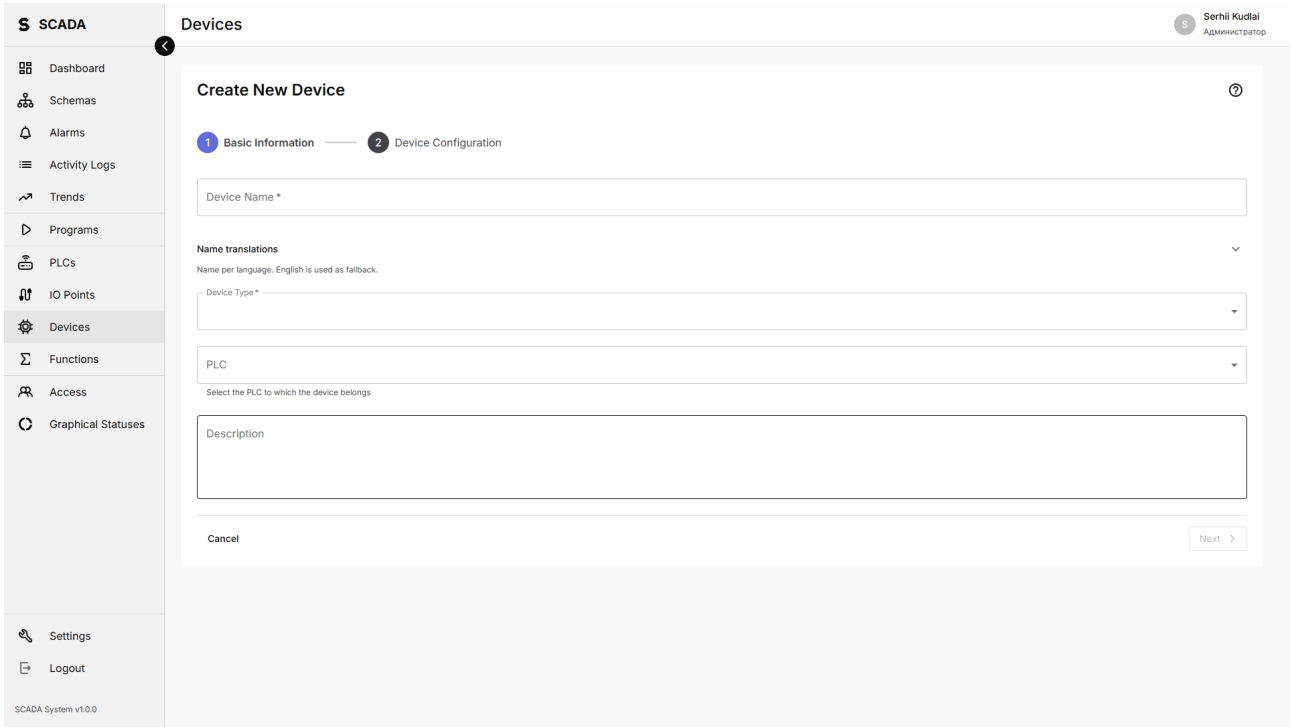
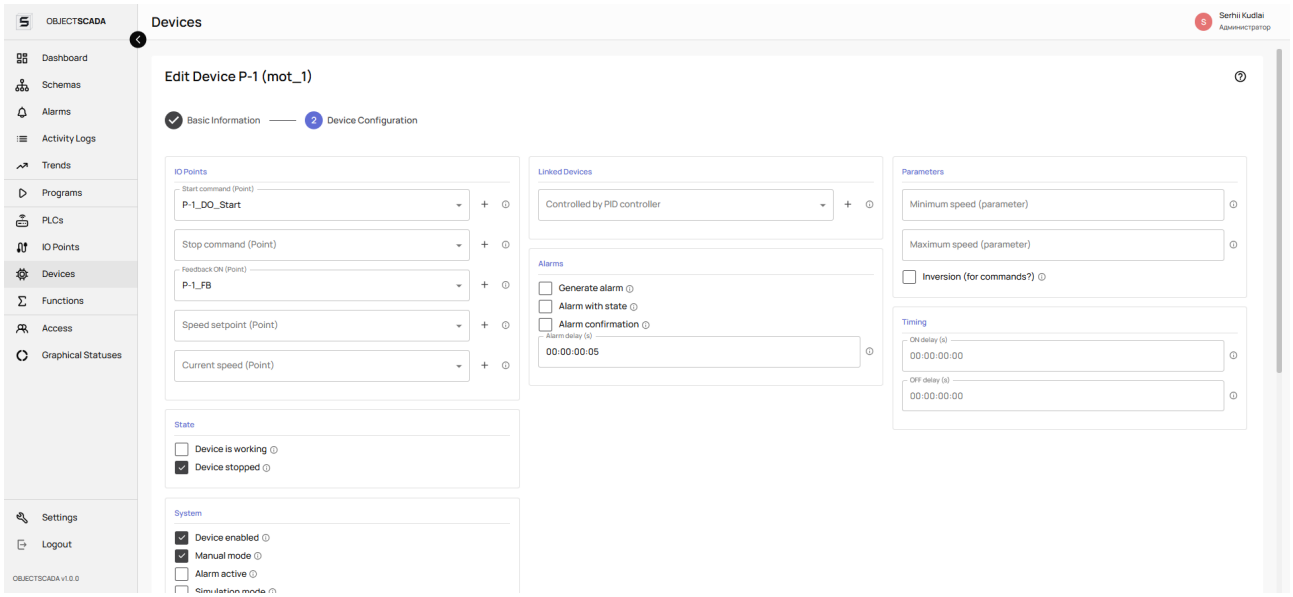


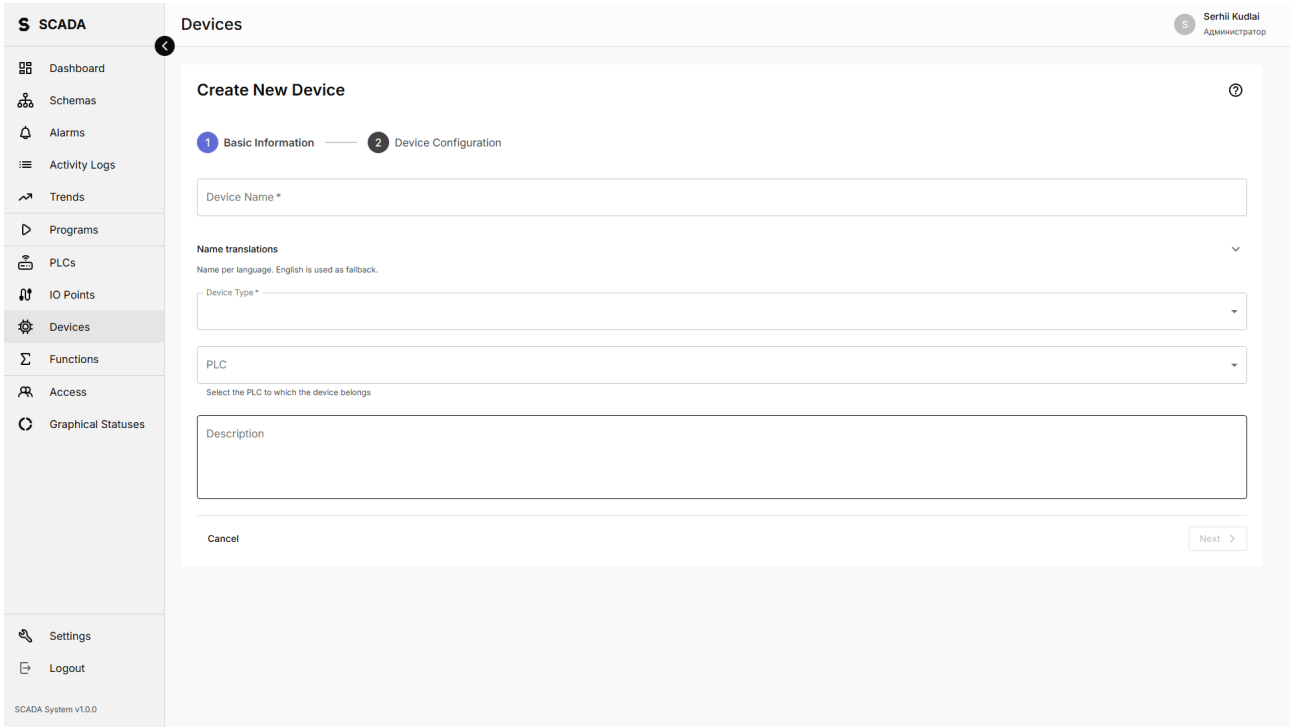
Table view: ID, Name, Type, Default value, Unit, Actions columns.



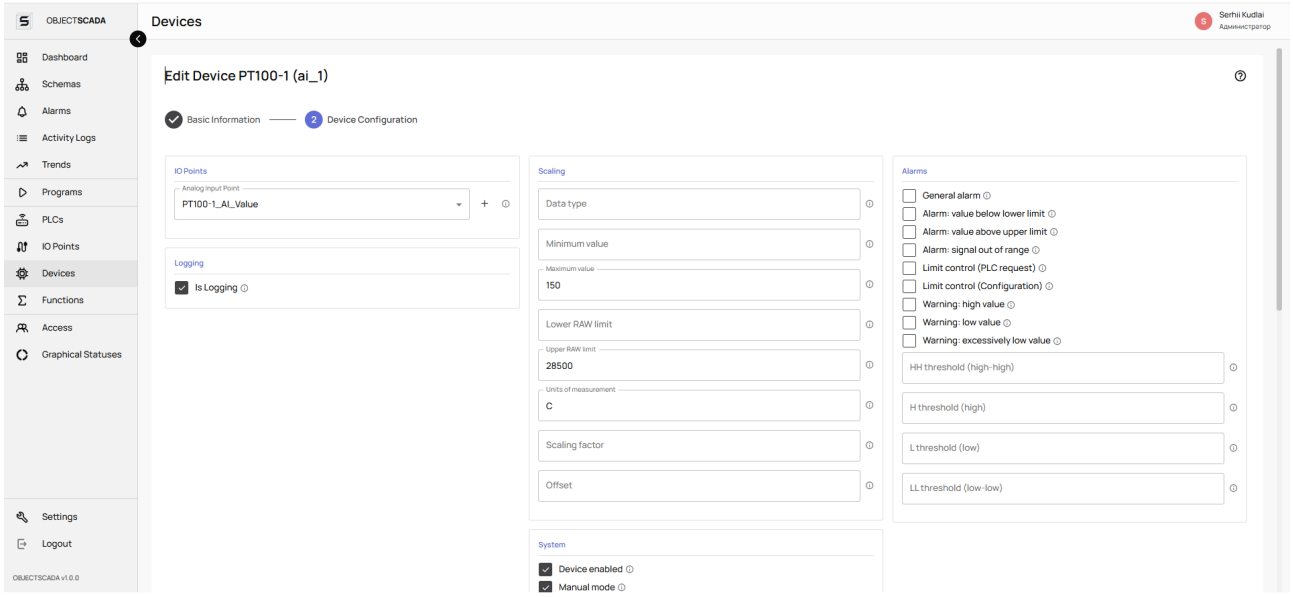
Motor creation form: primary fields — ID, Name, PLC, Description, device type.



Motor form: min/max speed, delays, generate_alarm, save buttons.



Analog input form: Min/Max, Units, Scaling, Raw Min/Max.



AI form: warning thresholds, default value and the rest of the fields.

Technical documentation

Device API (common to all types):

Method	Endpoint	Description
GET	/api/devices/all_devices/	All devices of all types
GET	/api/{device-type}/	List of devices of a type
POST	/api/{device-type}/	Create a device
GET	/api/{device-type}/{id}/	Retrieve a device
PUT	/api/{device-type}/{id}/	Update

Method	Endpoint	Description
PATCH	/api/{device-type}/{id}/	Partial update
DELETE	/api/{device-type}/{id}/	Delete

Endpoint types:

- /api/analog-inputs/
- /api/analog-outputs/
- /api/digital-inputs/
- /api/digital-outputs/
- /api/motors/
- /api/valves/
- /api/pid-controllers/
- /api/counters/
- /api/timers/
- /api/coordination-signals/
- /api/variable-speed-drives/

Additional actions (per type):

Method	Endpoint	Description
POST	/{id}/set_mode/	Toggle Auto/Manual
POST	/{id}/set_simulation/	Enable/disable simulation
POST	/{id}/set_interlock/	Set/clear interlock
POST	/{id}/update_status/	Update current status
POST	/{id}/set_manual_value/	Set manual value
POST	/{id}/trigger_websocket_update/	Force WS update (AllowAny)

WebSocket: Device update

Connection: ws:///ws/device/{device_id}

```
{
  "type": "device_status_update",
  "device": "ai_001",
  "data": {
    "current_status": {
      "value": 23.5,
      "mode": "auto",
      "simulation": false,
      "interlock": false,
      "quality": "good"
    },
    "status_icon": null,
    "updated_at": "2026-04-07T12:00:00Z"
  }
}
```

7. PLCs

For users

The PLC management page.

Screen elements:

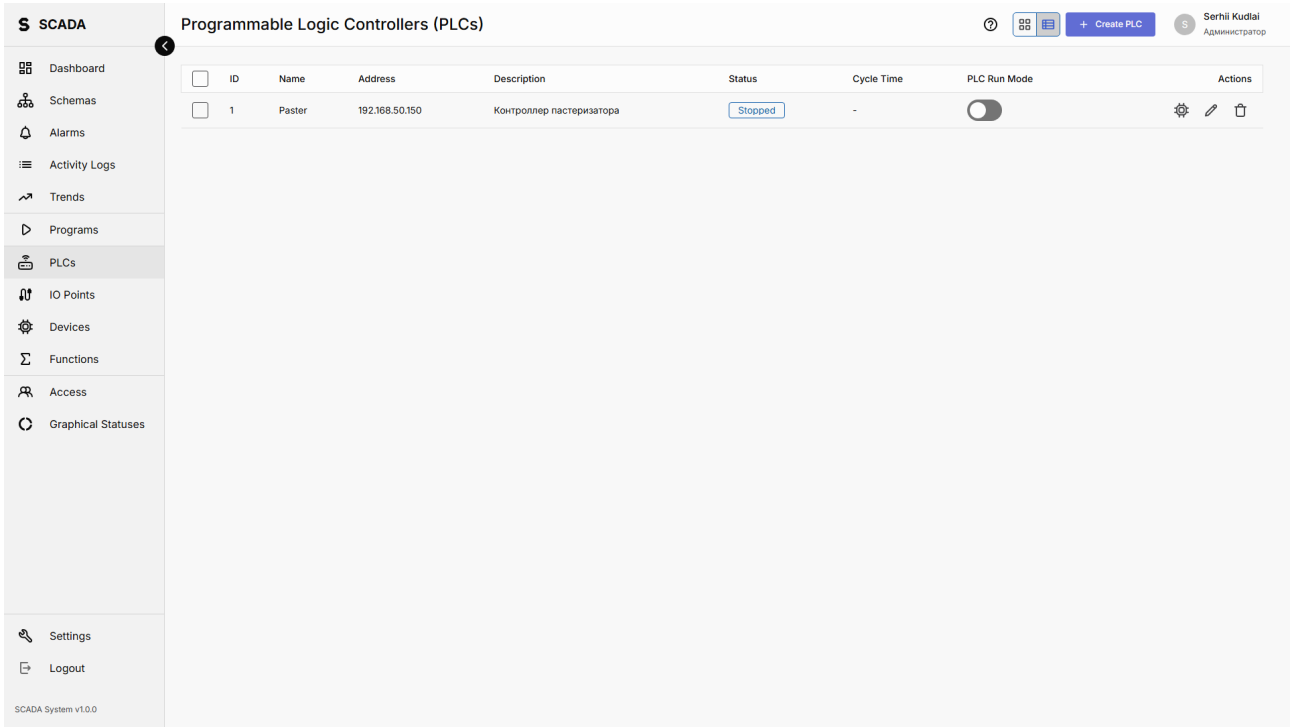
Element	Description
PLC list	Cards/table of PLCs
Status indicator	Green — online, grey — offline, red — error
«+» button	Add a new PLC
«Hardware Config» button	Go to the hardware configurator
Run Mode toggle	Enable/disable PLC run mode

Fields on a PLC card:

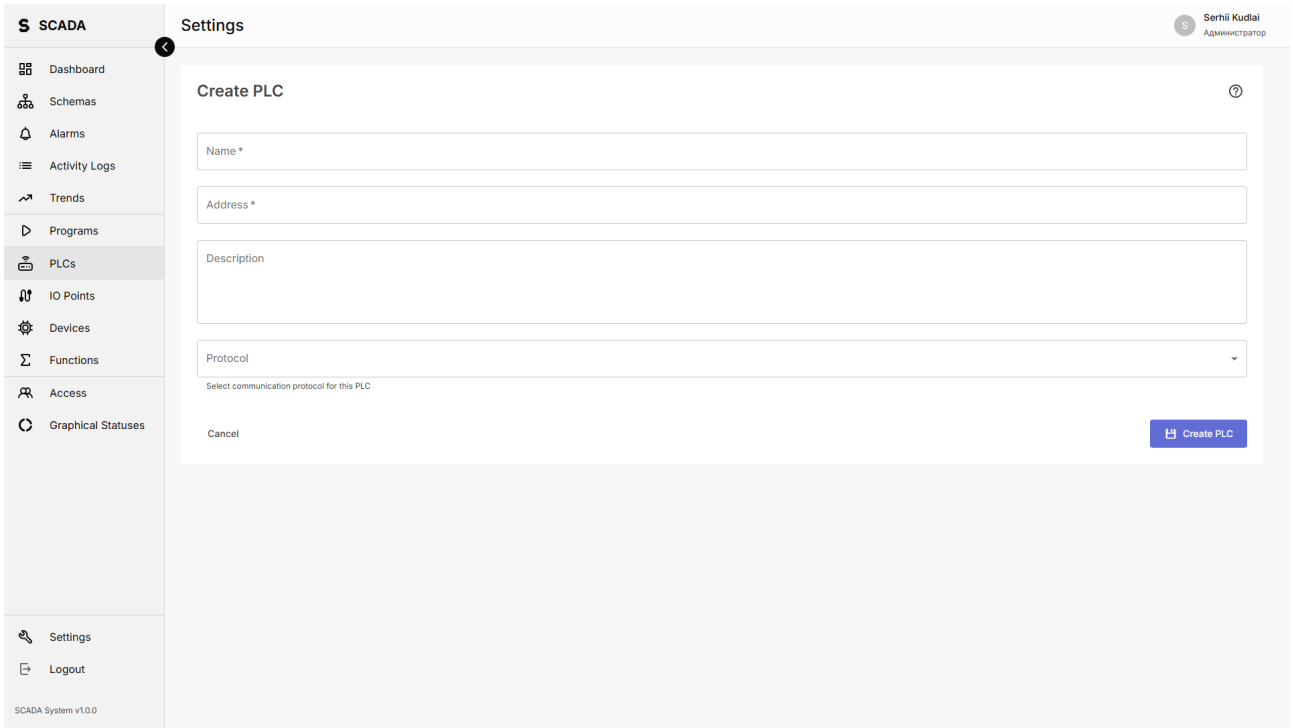
Field	Description
Name	PLC name
Protocol	Communication type (Siemens S7, Modbus, OPC UA)
Status	Online / Offline
Cycle time	Polling rate
IP address	Network address
Device count	Number of attached devices

PLC form:

Field	Description
Name	PLC name
Description	Text description
Protocol	Protocol selector
IP / Port	Network settings
Rack / Slot	For Siemens S7
«Test connection» button	Tests connectivity to the PLC
«Save» button	Saves the configuration



PLC list: cards with status indicators (green/grey), name, protocol, IP, Run Mode toggle, Hardware Config button.



PLC configuration form: Name field, Protocol dropdown, IP Address, Port, Rack, Slot, Test Connection button.

Technical documentation

PLC API:

Method	Endpoint	Description
GET	/api/plcs/	List PLCs
POST	/api/plcs/	Create a PLC

Method	Endpoint	Description
GET	/api/plcs/{id}/	Retrieve a PLC
PUT	/api/plcs/{id}/	Update
DELETE	/api/plcs/{id}/	Delete
GET	/api/plcs/{id}/status/	PLC status
POST	/api/plcs/{id}/set-run-mode/	Toggle Run Mode
GET	/api/plcs/status/	Status of all PLCs

PLC simulation:

Method	Endpoint	Description
POST	/api/plc/simulation/run/	Run simulation
GET	/api/plc/simulation/status/	Simulation status
POST	/api/plc/service/start/	Start PLC service
POST	/api/plc/service/stop/	Stop PLC service

8. Sequences

For users

Sequences define the order of operation for production equipment — the steps, transitions between them and switching conditions.

Screen elements:

Element	Description
Sequence list	Cards/table with filtering
«+» button	Create a new sequence
Status indicator	Green — running, grey — stopped, yellow — paused
Status filter	All / Running / Stopped
Search	Search by name

Sequence editor (tabs):

Tab	Description
Properties	Common settings (name, description, device bindings)
Steps	List of steps in execution order
Transition conditions	Configuration of transitions between steps
Events	Event handling
Accidents	Accident handling
Options	Options configuration
Recipe variables	Variable definitions
Production recipes	Recipes for this sequence
Pulse modules	Pulse-module control
Batches	Batch management

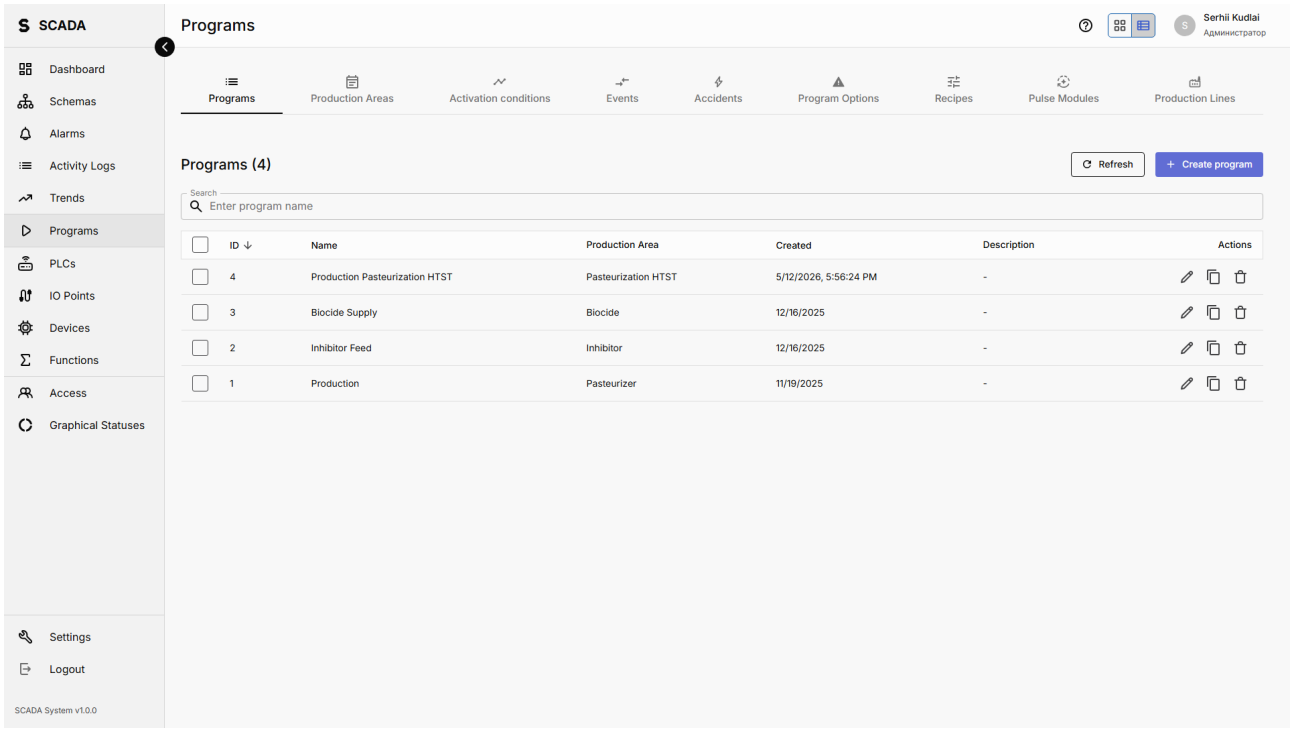
Step configuration:

Field	Description
Order	Step number in the sequence
Name	Step name (+ translations)
Description	Text description
Devices	Bound devices and their values at this step
Next steps	JSON map of transitions: {"target_step_order": [condition_id, ...]}
Batch action	NONE / TRANSFER / COMPLETE

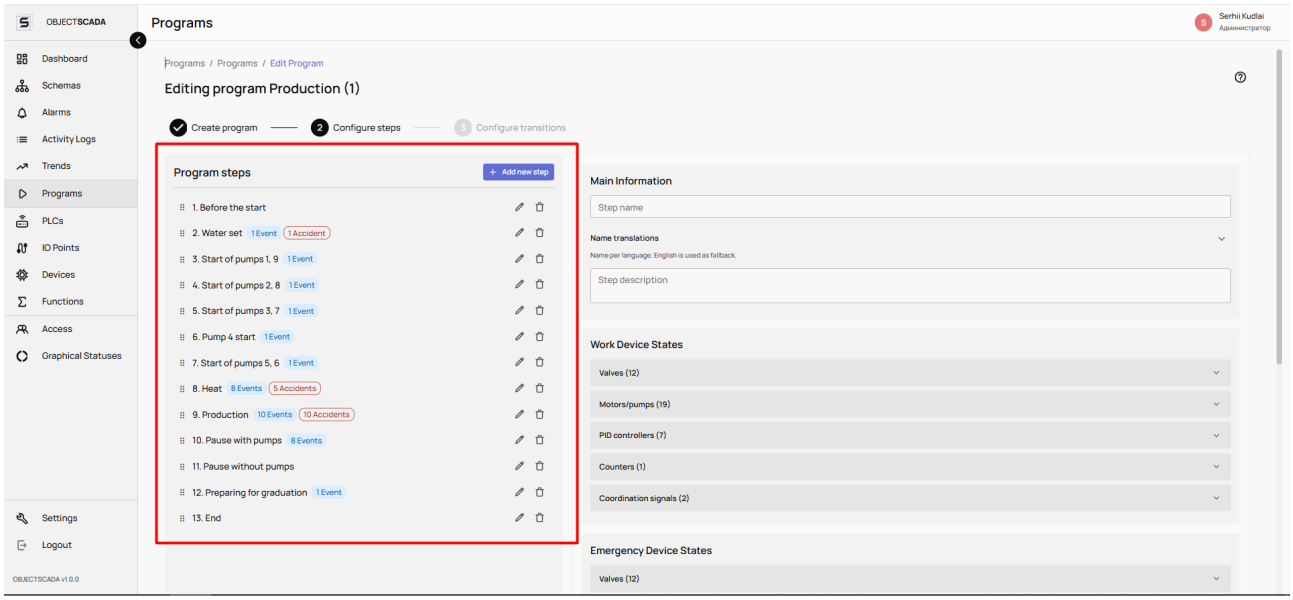
Transition condition:

Field	Description
Type	Device (by value), Timer (by time), Option (by operator action)
Device	Monitored device (for Device type)

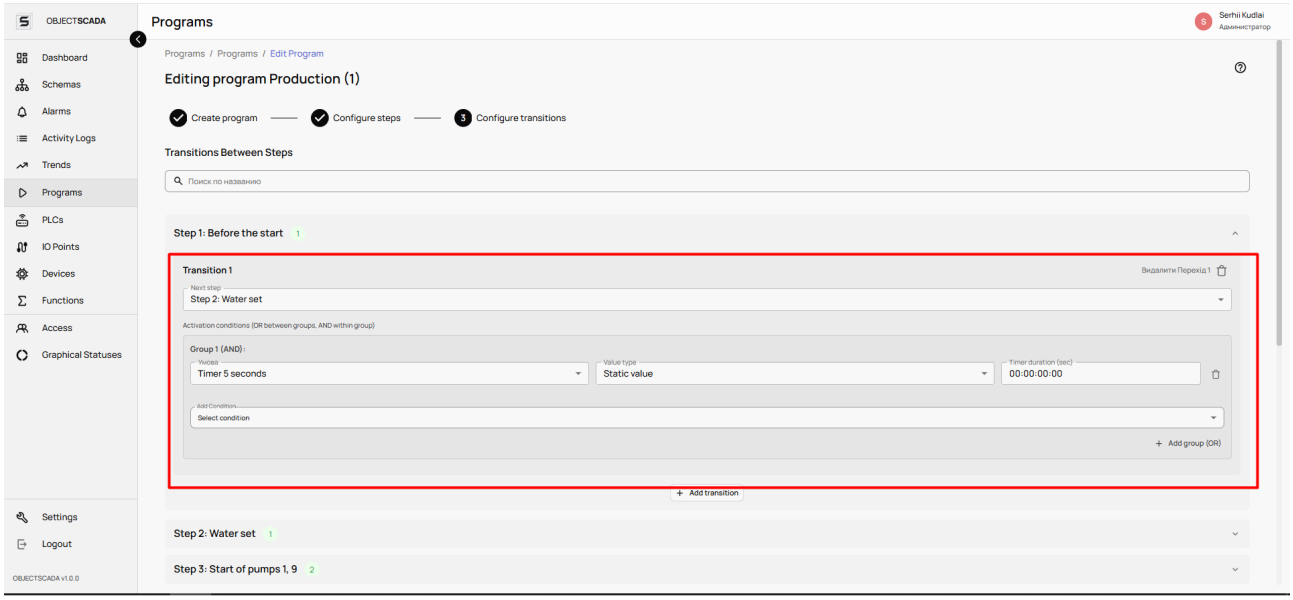
Field	Description
Operator	==, !=, >, =, <=
Value	Comparison value
Duration	Time (for Timer)
Simulation	Enable/disable condition simulation (for testing)



Sequences list: cards with name, status indicator (green/grey), current step, Edit/Delete buttons.



Step editor: numbered list of steps, each with name, bound devices, transition arrows, Add/Delete Step buttons.



Transition conditions editor: table with columns Type, Device, Operator, Value, Duration, Simulation toggle.

Create transition condition dialog: type selector (Device/Timer/Option), device dropdown, comparison operator, value field.

Technical documentation

Sequence API:

Method	Endpoint	Description
GET	/api/sequences/	List sequences
POST	/api/sequences/	Create
GET	/api/sequences/{id}/	Retrieve
PUT	/api/sequences/{id}/	Update
DELETE	/api/sequences/{id}/	Delete
GET	/api/sequences/{id}/steps/	List steps
POST	/api/sequences/{id}/add_step/	Add a step
POST	/api/sequences/{id}/set_steps_order/	Reorder steps

Method	Endpoint	Description
POST	/api/sequences/{id}/add_transition_conditions/	Add conditions

Step API:

Method	Endpoint	Description
GET	/api/sequence-steps/	All steps
POST	/api/sequence-steps/	Create a step
PUT	/api/sequence-steps/{id}/	Update
DELETE	/api/sequence-steps/{id}/	Delete

Transition conditions API:

Method	Endpoint	Description
GET	/api/transition-conditions/	All conditions
POST	/api/transition-conditions/	Create
PUT	/api/transition-conditions/{id}/	Update
DELETE	/api/transition-conditions/{id}/	Delete
POST	/api/transition-conditions/{id}/simulate/	Simulate condition
POST	/api/transition-conditions/clear_simulation/	Clear step simulation

WebSocket: Sequence status

Connection: ws:///ws/sequence/{sequence_id}

```
{
  "type": "sequence_status",
  "id": 1,
  "name": "Production A",
  "is_running": true,
  "is_pause": false,
  "is_stop": false,
  "active_step": 3,
  "active_step_data": {
    "id": 3,
    "order": 3,
    "name": "Heating",
    "name_translations": {"ru": "Нагрев"},
    "description": "..."
  },
  "previous_step": 2,
  "current_step_start_time": "2026-04-07T11:30:00Z",
  "monitored_devices": {
    "ai_001": {"value": 72.3, "mode": "auto"}
  },
  "activated_devices": {
    "mot_001": {"value": 1, "is_running": true}
  },
  "conditions": [
    {
      "id": 10,
      "type": "device",
      "device_id": "ai_001",
      "operator": ">=",
      "value": 80.0,
```

... (+9 строк)

9. Recipes and variables

For users

Recipes define the specific device parameter values at each step of a sequence. Recipe variables let you parameterise these values.

Recipe types:

Type	Description
Recipe (Step recipe)	Defines device values for each step of a sequence
Production Recipe	A set of variables for a specific production job

Step-recipe form:

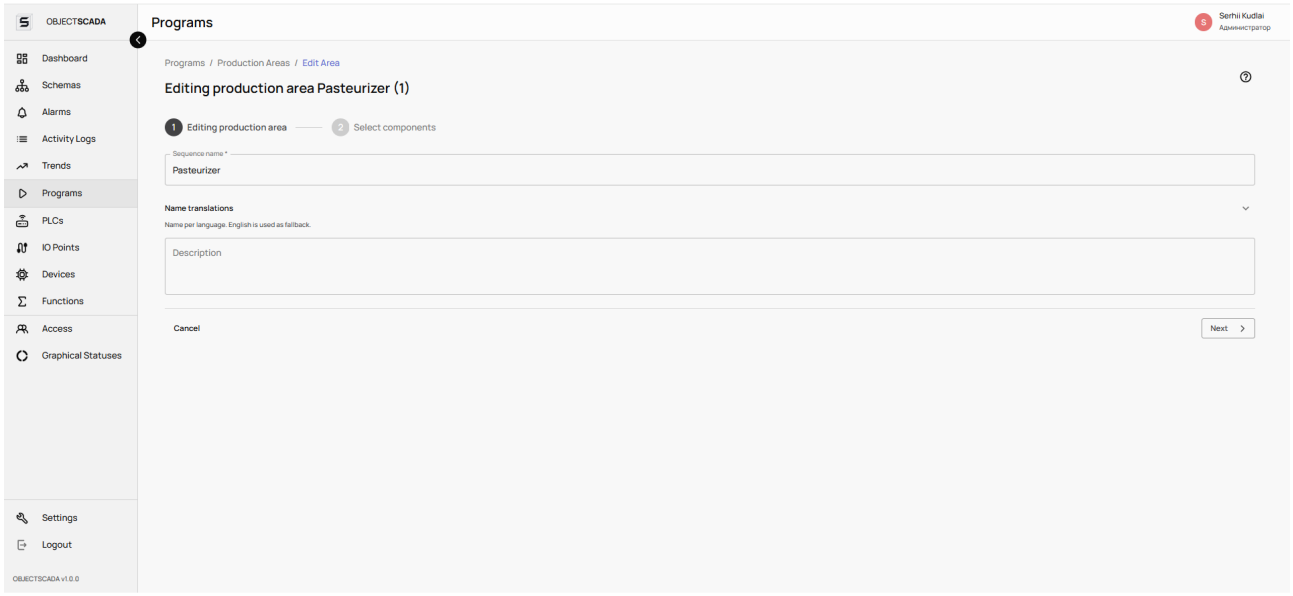
Field	Description
Name	Recipe name (+ translations)
Sequence	Bind to a sequence
Steps	Per step — a set of «device → value» entries

Production-recipe form:

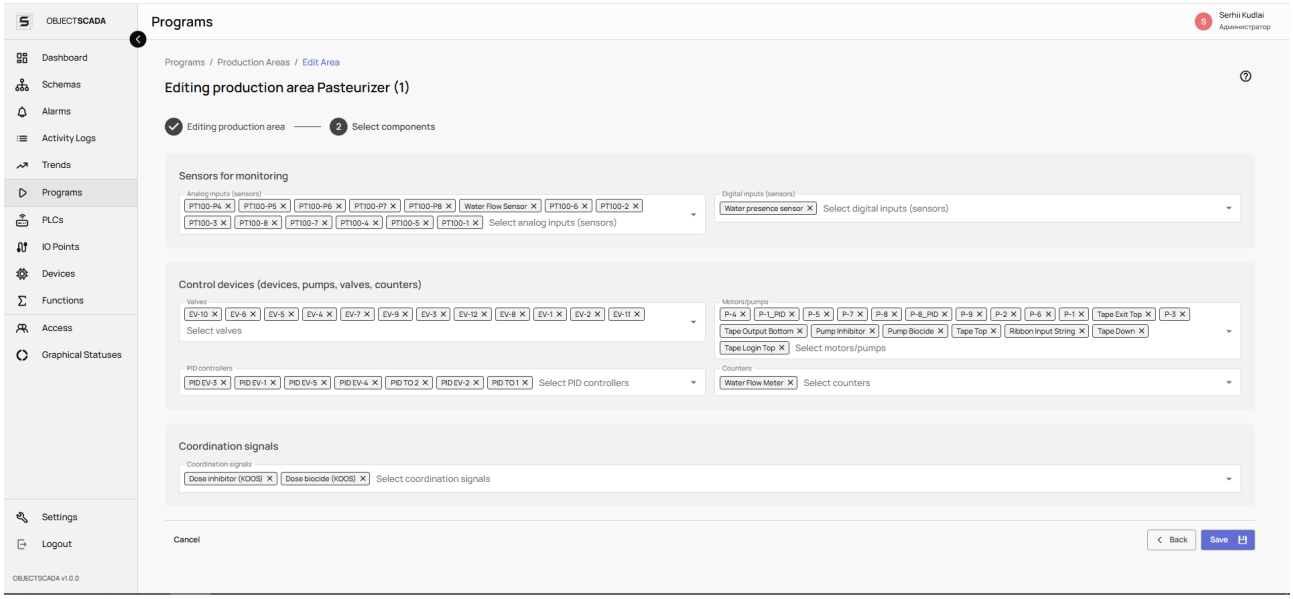
Field	Description
Name	Recipe name
Binding	To a sequence OR to a production line (XOR)
Variables	List of variables and their values

Recipe variables:

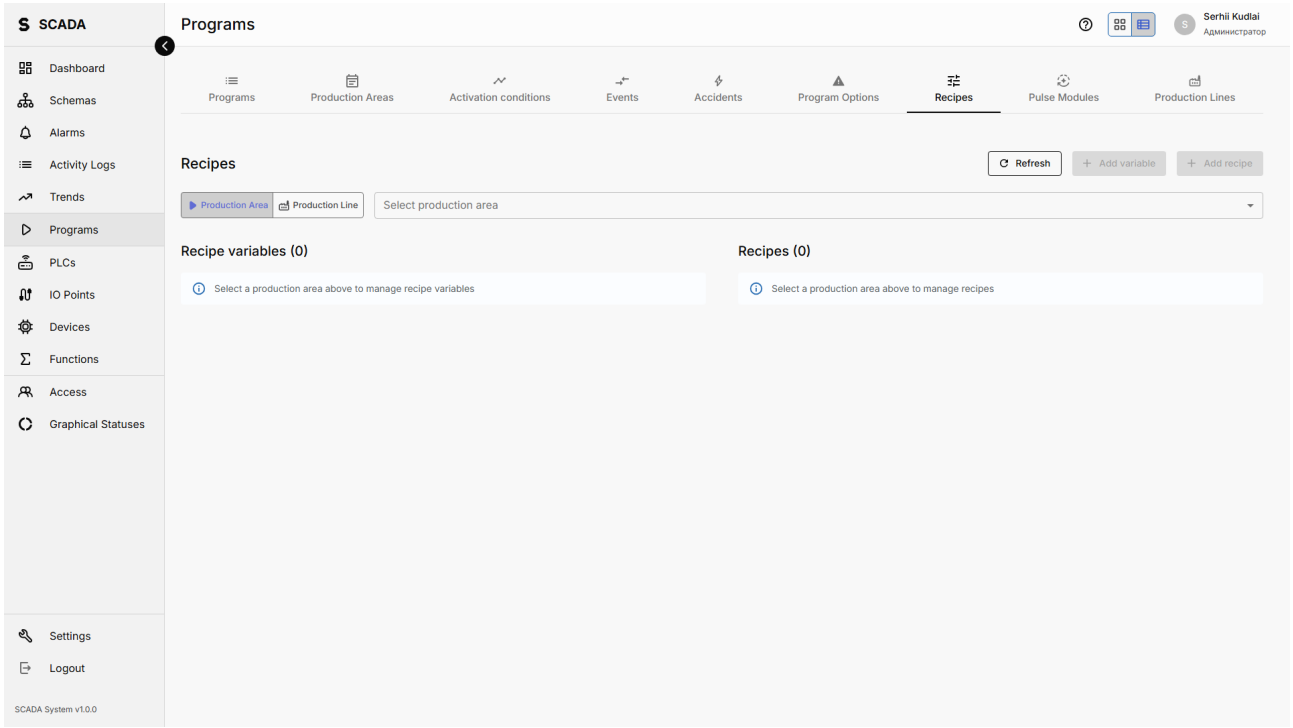
Field	Description
Name	Variable name (+ translations)
Binding	To a sequence OR to a production line (XOR)
Type	Numeric, textual
Default value	Initial value
Min / Max	Limits



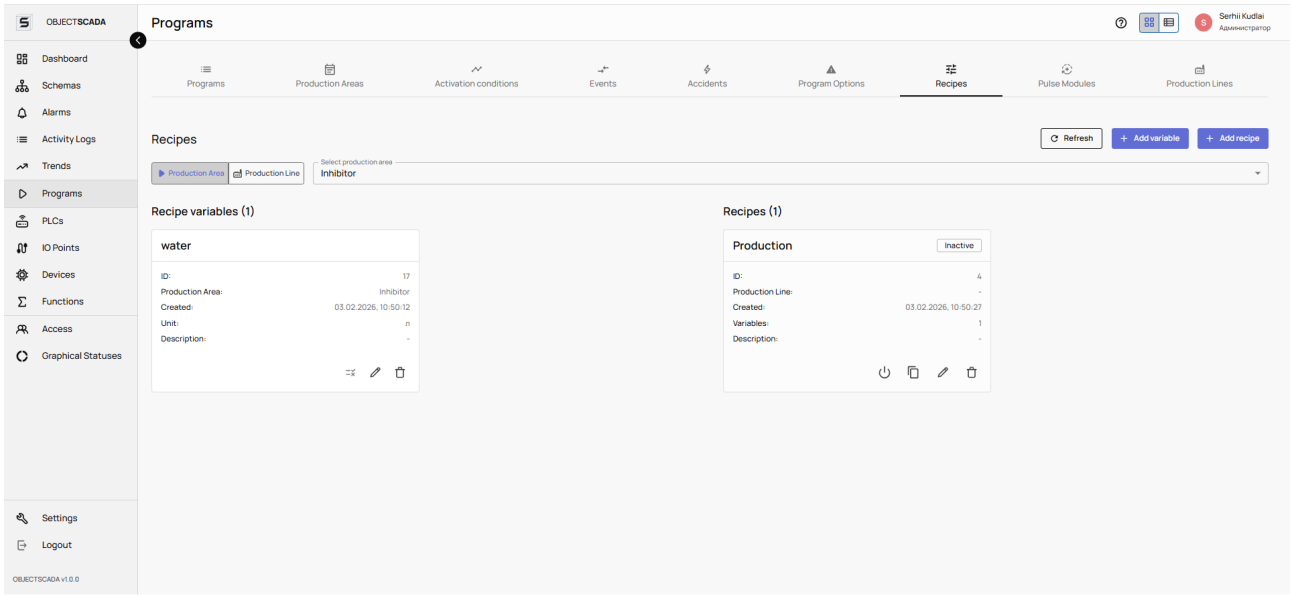
Recipe creation wizard — step 1: name and sequence binding.



Recipe creation wizard — step 2: step → device → value table.



Production recipes management: table of recipes with Create/Edit/Delete/Duplicate buttons, Sequence/Production Line toggle.



Variables management: list of variables with name, type, default value, Min/Max. Add/Delete buttons.

Technical documentation

Recipes API:

Method	Endpoint	Description
GET	/api/recipes/	List of step recipes
POST	/api/recipes/	Create a recipe
PUT	/api/recipes/{id}/	Update
DELETE	/api/recipes/{id}/	Delete

Production recipes API:

Method	Endpoint	Description	Parameters
GET	/api/production-recipes/	All recipes	?sequence=ID or ?production_line=ID
POST	/api/production-recipes/	Create	—
PUT	/api/production-recipes/{id}/	Update	—
DELET	/api/production-recipes/{id}/	Delete	—
POST	/api/production-recipes/{id}/select-production-recipe/	Activate	—

Variables API:

Method	Endpoint	Description	Parameters
GET	/api/production-recipe-variables/	All variables	?sequence=ID or ?production_line=ID
POST	/api/production-recipe-variables/	Create	—
PUT	/api/production-recipe-variables/{id}/	Update	—
DELET E	/api/production-recipe-variables/{id}/	Delete	—

10. Events and accidents

For users

Events — conditions that run actions during sequence operation (e.g. switching when a value is reached).

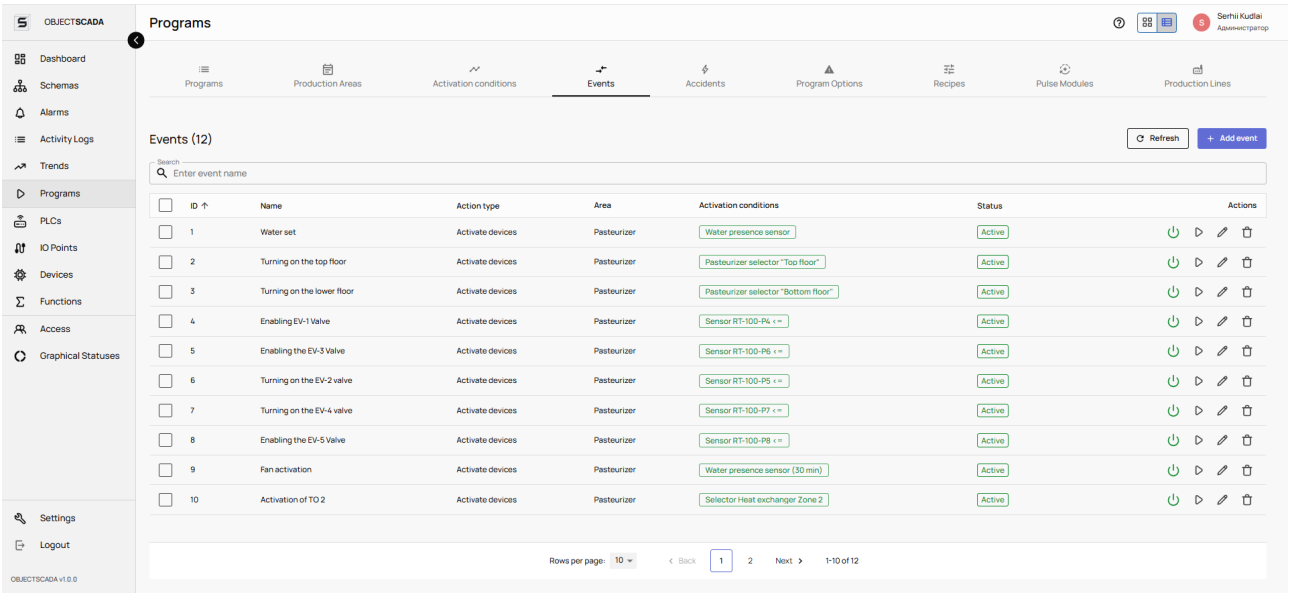
Accidents — critical conditions that halt the process when a dangerous situation arises.

Event form:

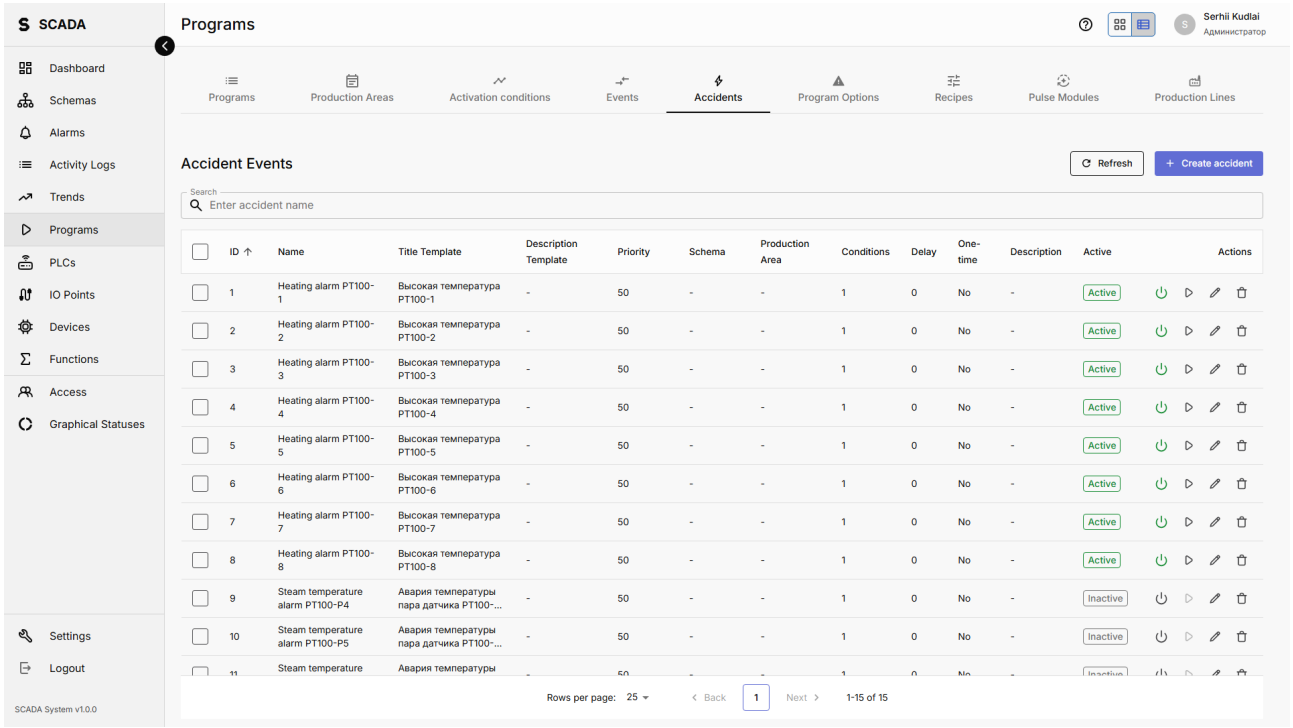
Field	Description
Name	Event name (+ translations)
Type	Trigger type
Device	Monitored device
Condition	Comparison operator and value
Action	What to execute on trigger

Accident form:

Field	Description
Name	Accident name (+ translations)
Severity	INFO / WARNING / ERROR / CRITICAL
Device	Monitored device
Condition	Operator and value
Interlock	Devices to lock out



Events management: table of events with Name, Device, Condition, Action columns, Add/Edit/Delete buttons.



Accidents management: table with severity colour coding (INFO blue, WARNING yellow, ERROR red, CRITICAL dark red).

Technical documentation

Events API:

Method	Endpoint	Description
GET	/api/events/	List of events
POST	/api/events/	Create
PUT	/api/events/{id}/	Update
DELETE	/api/events/{id}/	Delete
POST	/api/events/{id}/set_active/	Activate/deactivate

Accidents API:

Method	Endpoint	Description
GET	/api/accident-events/	List accident definitions
POST	/api/accident-events/	Create a definition
GET	/api/accidents/	List fired accidents
POST	/api/accidents/{id}/acknowledge/	Acknowledge
POST	/api/accidents/{id}/resolve/	Resolve
POST	/api/accidents/resolve-all/	Resolve all
GET	/api/accidents/active_events/	Currently active accidents

WebSocket: Accidents

Connection: ws:///ws/accidents

```
{
  "type": "accident_created",
  "accident_id": 42,
  "title": "High temperature",
  "description": "AI_001 exceeded the 95°C threshold",
  "status": "NEW",
  "severity_level": "CRITICAL",
  "device_id": "ai_001",
  "device_name": "Temperature Sensor 1",
  "schema_id": 1,
  "occurred_at": "2026-04-07T12:00:00Z"
}
```

11. Options

For users

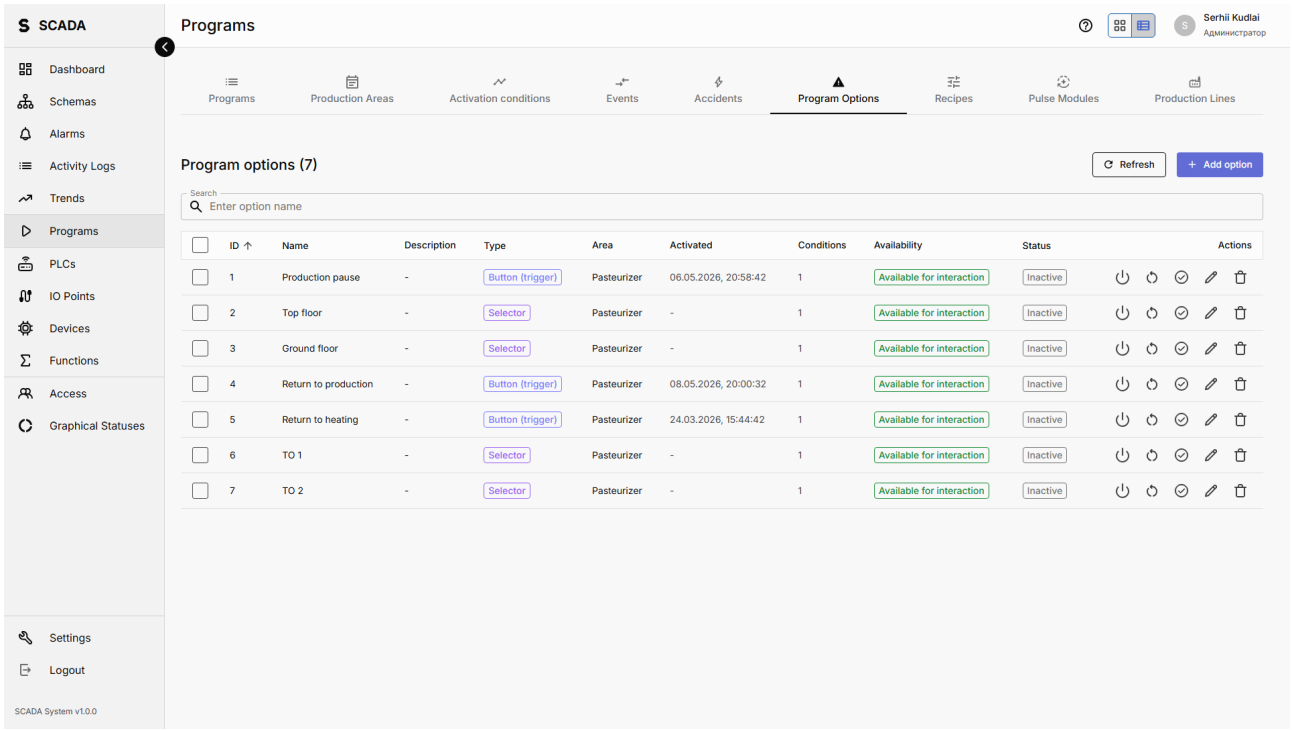
Options are buttons/switches available to the operator during sequence execution. They let the operator select alternative execution paths.

Option form:

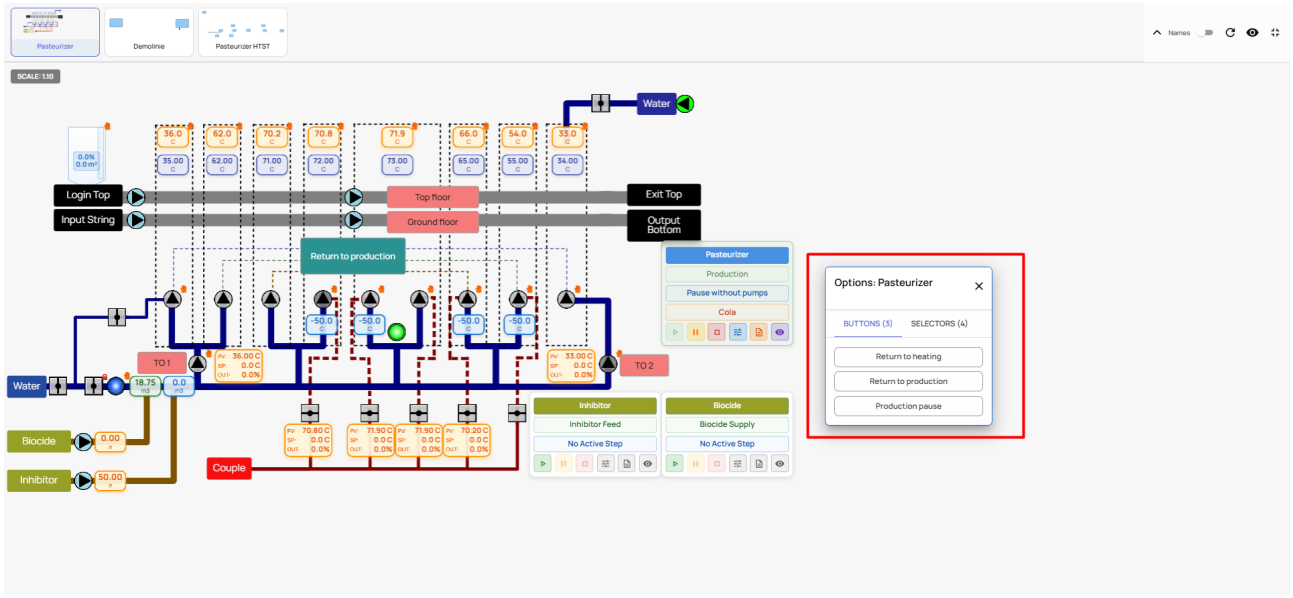
Field	Description
Name	Option name (+ translations)
Sequence	Sequence binding
Description	Text description

On the schema (Schema Viewer):

- Options appear as buttons
- Pressing toggles the option (on/off)
- The button colour changes with state



Options management: list of options with name, sequence, description, Edit/Delete buttons.



Option button on a schema: rectangular button with a name, green — active, grey — inactive.

Technical documentation

Options API:

Method	Endpoint	Description
GET	/api/sequence-options/	List of options
POST	/api/sequence-options/	Create
PUT	/api/sequence-options/{id}/	Update
DELETE	/api/sequence-options/{id}/	Delete

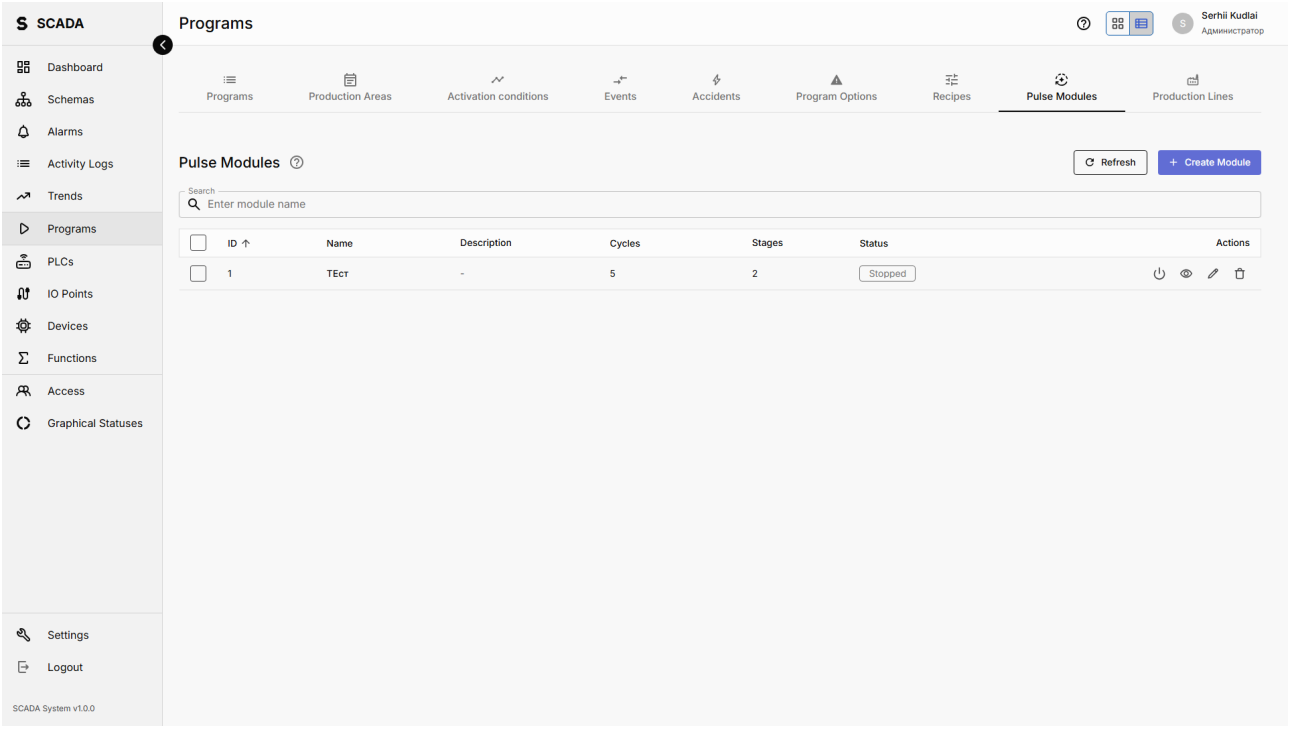
12. Pulse modules

For users

Pulse modules generate cyclic on/off device sequences (for example, cyclic water supply).

Structure:

Level	Description
Module	Container of steps
Step	One stage of a cycle
Phase	A time interval inside a step (on/off)
Phase device	Which device is switched on/off



Pulse-module management: hierarchical list — module → steps → phases → devices. Add/Edit/Delete buttons on every level.

Technical documentation

Pulse modules API:

Method	Endpoint	Description
GET	/api/pulse-modules/	List modules
POST	/api/pulse-modules/	Create
POST	/api/pulse-modules/{id}/start/	Start
POST	/api/pulse-modules/{id}/stop/	Stop
GET	/api/pulse-module-steps/	Module steps
GET	/api/pulse-module-step-phases/	Step phases

Method	Endpoint	Description
GET	/api/pulse-module-devices/	Phase devices

13. Batch management

For users

The batch management system tracks production batches across the whole process.

Batch panel elements:

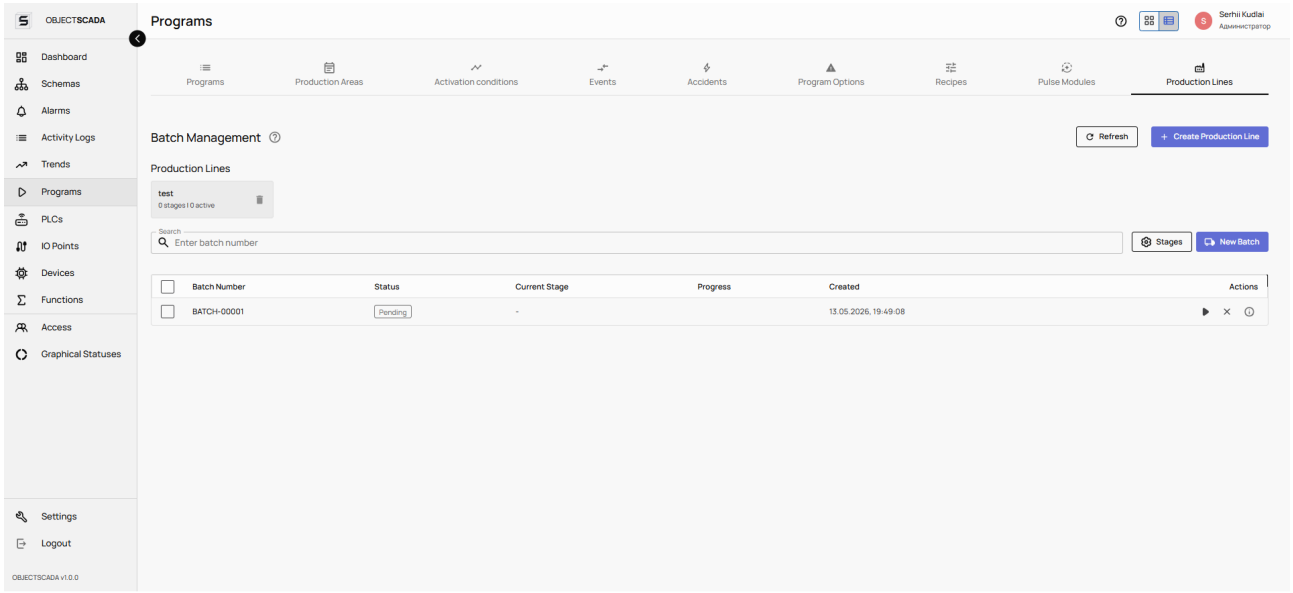
Element	Description
Create batch	Button to create a new batch
Batch table	List with number, recipe, status, progress
Batch status	Pending / Queued / In Progress / Paused / Completed / Failed / Cancelled
Queue	Processing order

Batch actions:

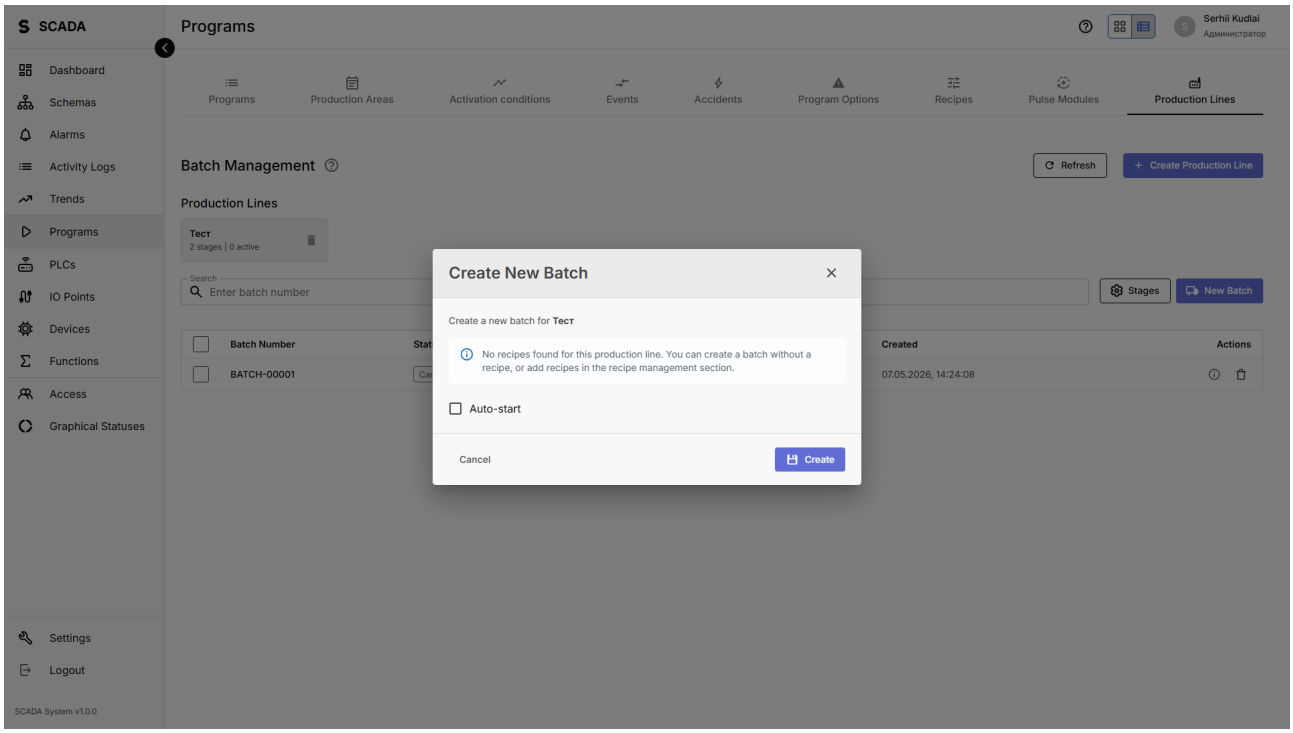
Button	Description
Play	Start/resume a batch
Pause	Pause
Stop	Full stop
Skip	Skip the current step
Cancel	Cancel the batch
Delete	Remove from the list

Create-batch dialog:

Field	Description
Production line	Line selector
Recipe	Recipe selector
Batch number	Unique number (BAT-001)
Auto-start	Start immediately after creation



Batch management panel: table of batches with status colour coding (green — In Progress, grey — Pending, blue — Completed, red — Failed), control buttons.



Create dialog: line and recipe selectors, number field, «Auto-start» checkbox, «Create» and «Cancel» buttons.

Technical documentation

Batches API:

Method	Endpoint	Description
GET	/api/batches/	All batches
POST	/api/production-lines/{id}/create_batch/	Create a batch
POST	/api/batches/{id}/start/	Start
POST	/api/batches/{id}/pause/	Pause

Method	Endpoint	Description
POST	/api/batches/{id}/resume/	Resume
POST	/api/batches/{id}/cancel/	Cancel
POST	/api/batches/{id}/transfer/	Transfer to the next stage
GET	/api/batches/{id}/history/	Stage history
GET	/api/batch-queue/	Batch queue
POST	/api/batch-queue/reorder/	Reorder the queue

WebSocket: Batch status

Connection: ws:///ws/schema/{schemaId}/batches

```
{
  "type": "batch_status",
  "batch_id": "uuid-...",
  "batch_number": "BAT-001",
  "status": "IN_PROGRESS",
  "current_stage": "Mixing",
  "progress": 65,
  "production_line_id": "uuid-...",
  "recipe_name": "Recipe A",
  "updated_at": "2026-04-07T12:00:00Z"
}
```

WebSocket commands (Client → Server):

```
{
  "type": "batch_command",
  "command": "start",
  "batch_id": "uuid-...",
  "production_line_id": "uuid-..."
}
```

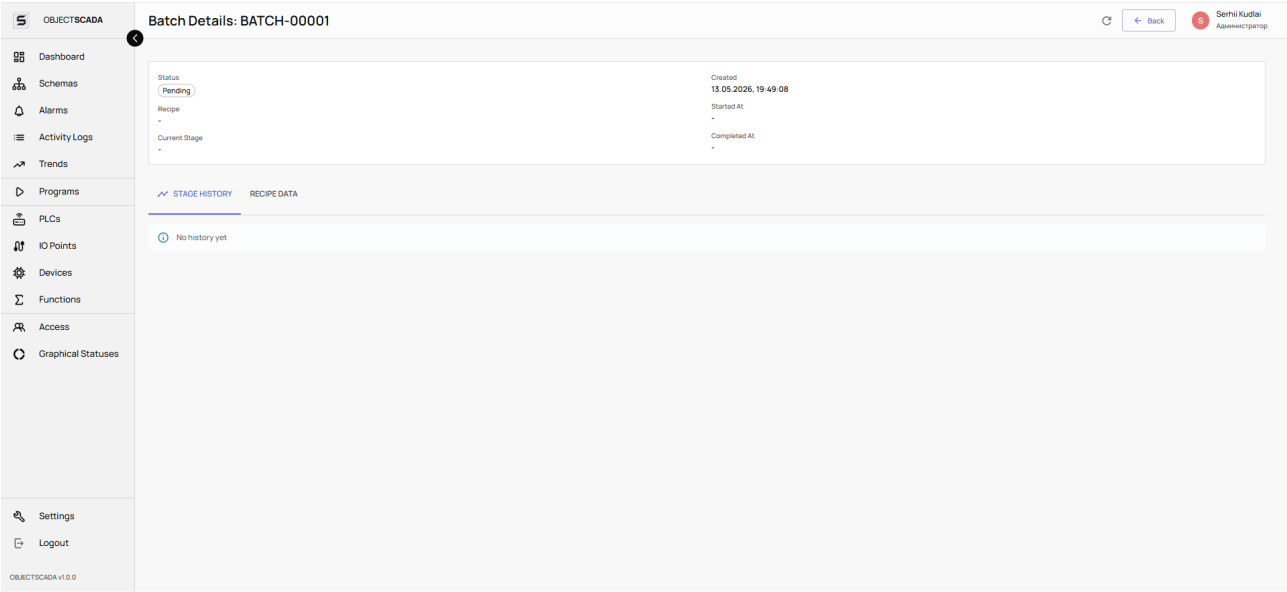
14. Production lines

For users

A production line ties several sequences together into a single production route with stages.

Fields:

Field	Description
Name	Line name (+ translations)
Stages	List of Production Line Stages, each bound to a sequence
Stage order	Defines the batch route
Status	Active / Inactive



Lines management: table with name, stage count, status, Edit/Delete buttons. The detail view — list of stages with drag-and-drop reordering.

Technical documentation

Production lines API:

Method	Endpoint	Description
GET	/api/production-lines/	List lines
POST	/api/production-lines/	Create
PATCH	/api/production-lines/{id}/	Update
DELETE	/api/production-lines/{id}/	Delete
POST	/api/production-lines/{id}/activate/	Activate
POST	/api/production-lines/{id}/deactivate/	Deactivate
GET	/api/production-line-stages/	All stages
POST	/api/production-line-stages/	Create a stage
PATCH	/api/production-line-stages/{id}/	Update

Method	Endpoint	Description
DELETE	/api/production-line-stages/{id}/	Delete

15. Trends

For users

Trends show historical device data as charts.

Screen elements:

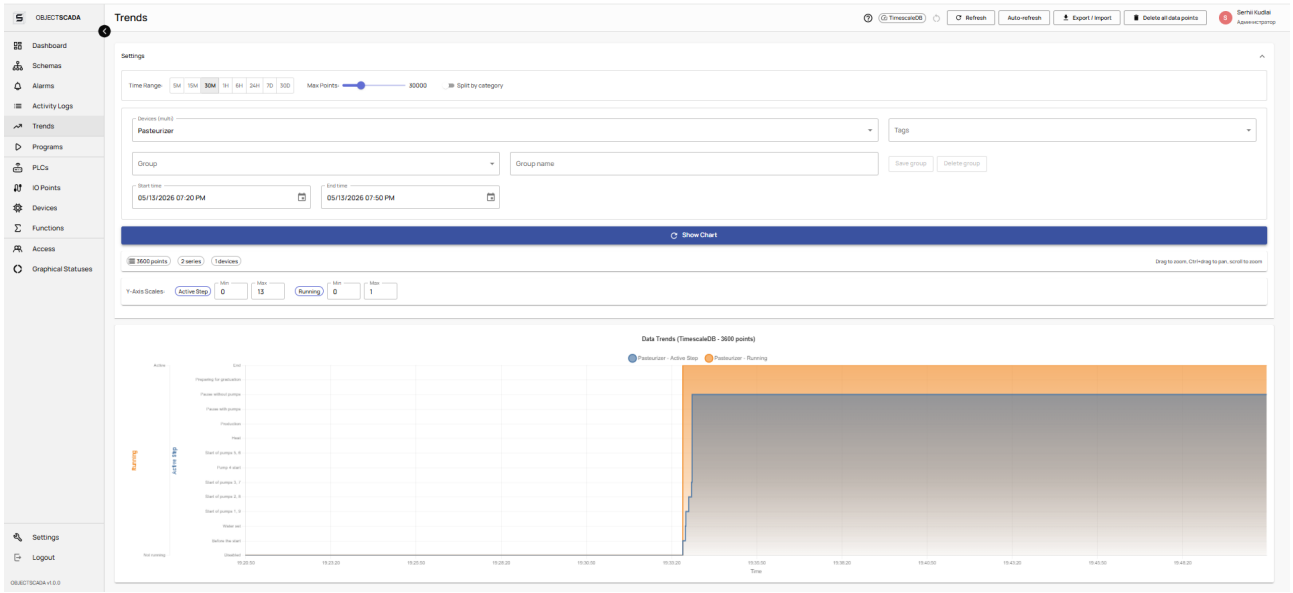
Element	Description
Device selector	Multi-select of devices/parameters for the chart
Time range	Presets: 5 min, 15 min, 30 min, 1 h, 6 h, 24 h, 7 d, 30 d
Custom period	Free date/time range
Chart	Chart.js line chart with zooming
Device groups	Save/load device sets
Storage	Disk usage information

Chart interactions:

Action	Description
Drag	Select a region to zoom in
Scroll (mouse wheel)	Zoom
Right-click drag	Pan
Hover	Tooltip with values
Reset zoom	Reset to the original scale

Buttons:

Button	Description
Add devices	Opens device multi-select
Save group	Saves the current device set
Load group	Pick a saved group
Delete group	Delete a saved group
Download	Export data (CSV/JSON)
Download chart	Export chart as an image
Delete data	Delete data points (superuser)
Refresh	Force data reload



Full Trends view: top — device multi-select (chips), time range buttons (5m/15m/1h/6h/24h/7d/30d) and a DateTimePicker. Centre — line chart with multiple colour lines. Right — legend.

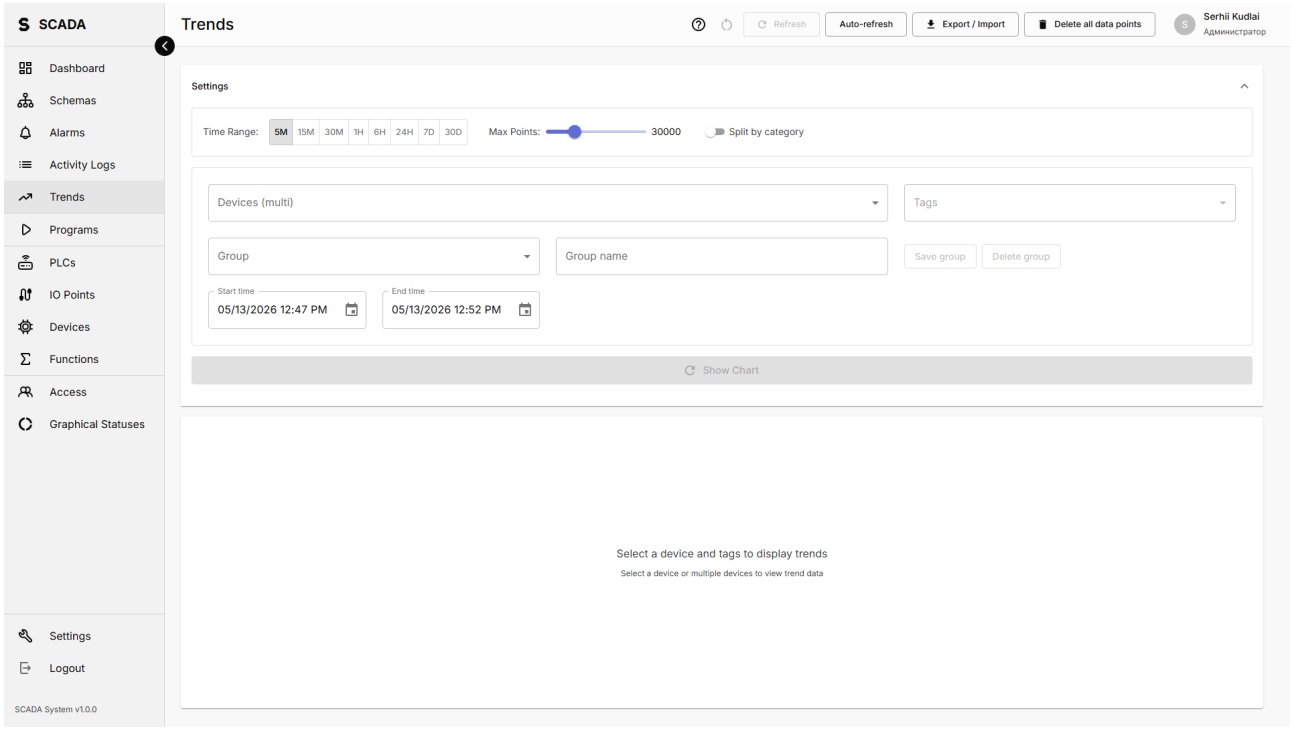
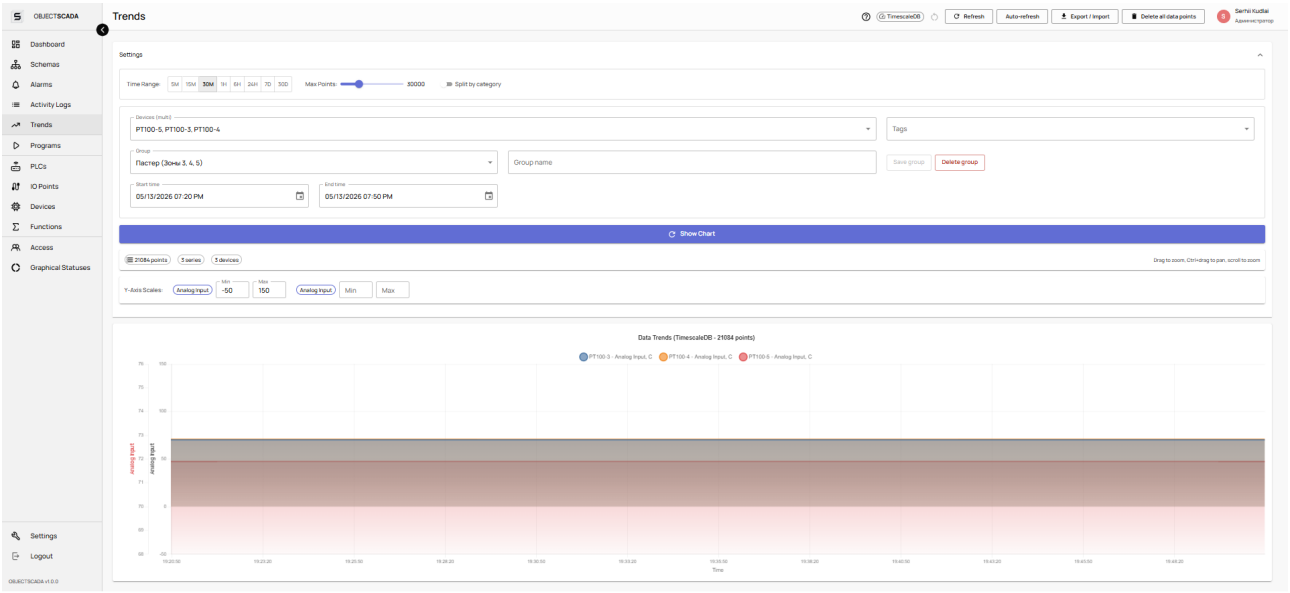


Chart after zoom: selected region zoomed in, Reset zoom button visible.



Saved device groups dropdown: «Group 1 — Temperatures», «Group 2 — Pressures», Save/Load/Delete buttons.

Technical documentation

Trends API (TimescaleDB):

Method	Endpoint	Description	Parameters
GET	/api/graphs/graph/	Chart data	?start_time=ISO&end_time=ISO&device_ids=id1,id2
GET	/api/graphs/available-parameters/	Available parameters	—
GET	/api/graphs/timescale-status/	TimescaleDB status	—
DELET	/api/graphs/delete-all/	Delete all data	—
GET	/api/data-points/	Data points	?limit=100&offset=0

Export data:

Method	Endpoint	Description
POST	/api/project/export-graph-data/	Export charts
POST	/api/project/import-graph-data/	Import charts
GET	/api/project/graph-data-stats/	Data statistics

16. Alarms

For users

Centralised journal of all alarm events in the system.

Table columns:

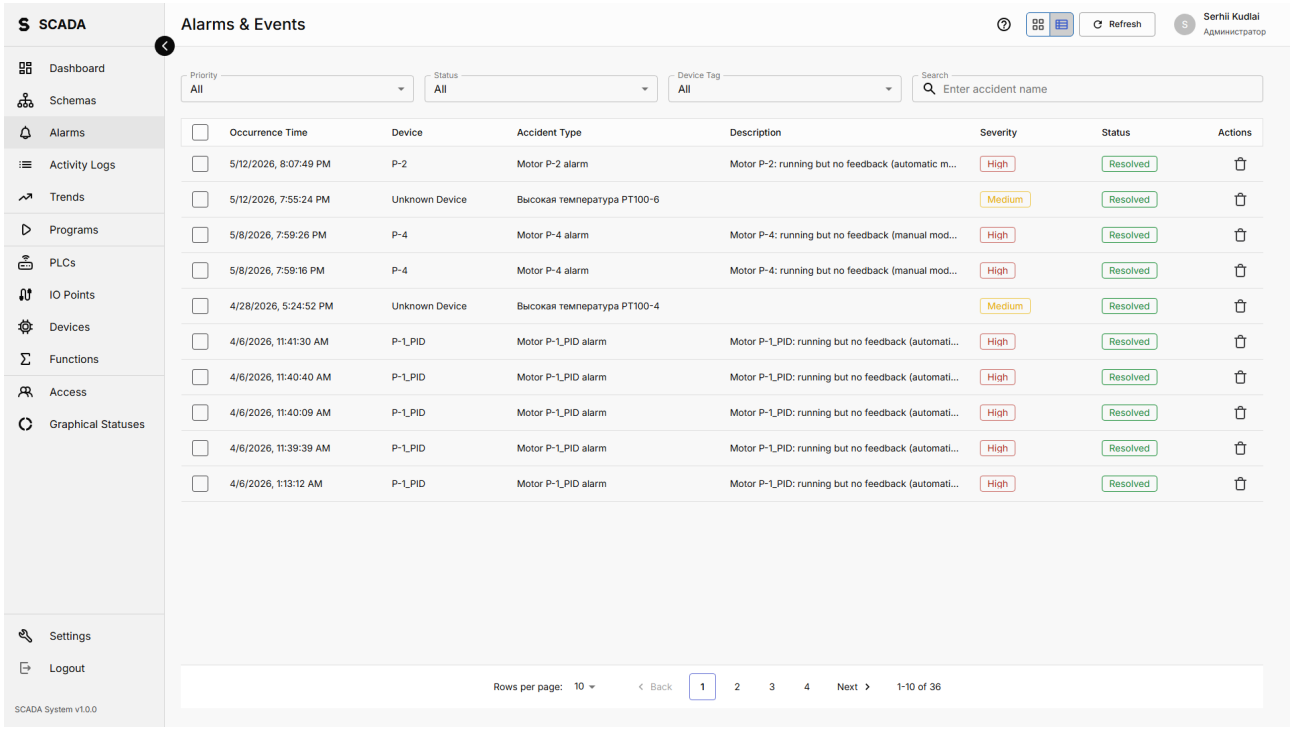
Column	Description
Time	Occurrence date and time
Severity	Error (red) / Warning (yellow)
Message	Alarm description
Device	Source
Status	Active / Acknowledged / Resolved
Actions	Control buttons

Filters:

Filter	Description
By severity	Warning / Error / All
By status	Active / Acknowledged / Resolved / All
By device	Device selector
Search	Text search

Buttons:

Button	Description
Acknowledge	Acknowledge (take note)
Resolve	Resolve (problem fixed)
Resolve All	Resolve all alarms
Delete	Remove an entry
Refresh	Reload the list



Alarms page: table with severity colour coding (red left stripe — Error, yellow — Warning). Filters on top. Acknowledge/Resolve buttons in each row.

Technical documentation

Accidents API: see section 10.

WebSocket: ws/accidents — receives new accidents and status updates in real time.

17. Math functions

For users

Math functions let you compute derived values from device data and recipe variables.

Screen elements:

Element	Description
Function list	Table/cards of functions
«+» button	Create a function
Search	By name

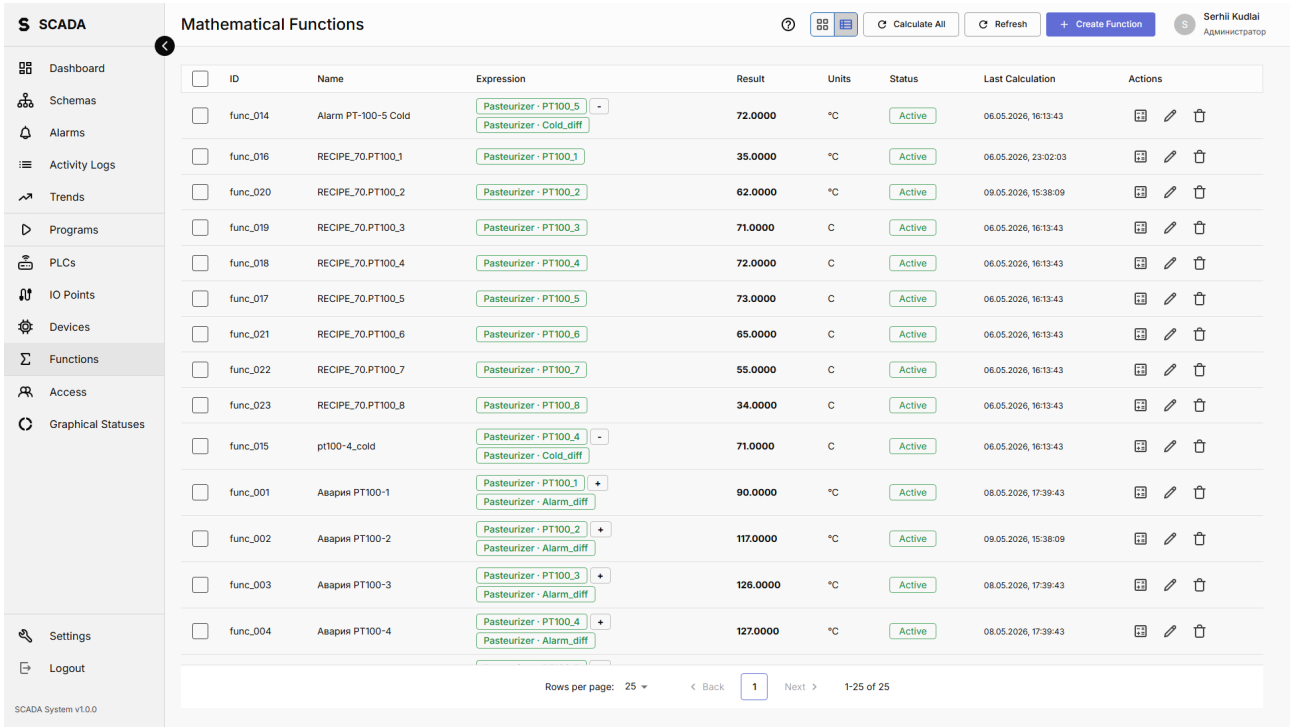
Expression editor (block-based):

An expression is built up from blocks added in sequence:

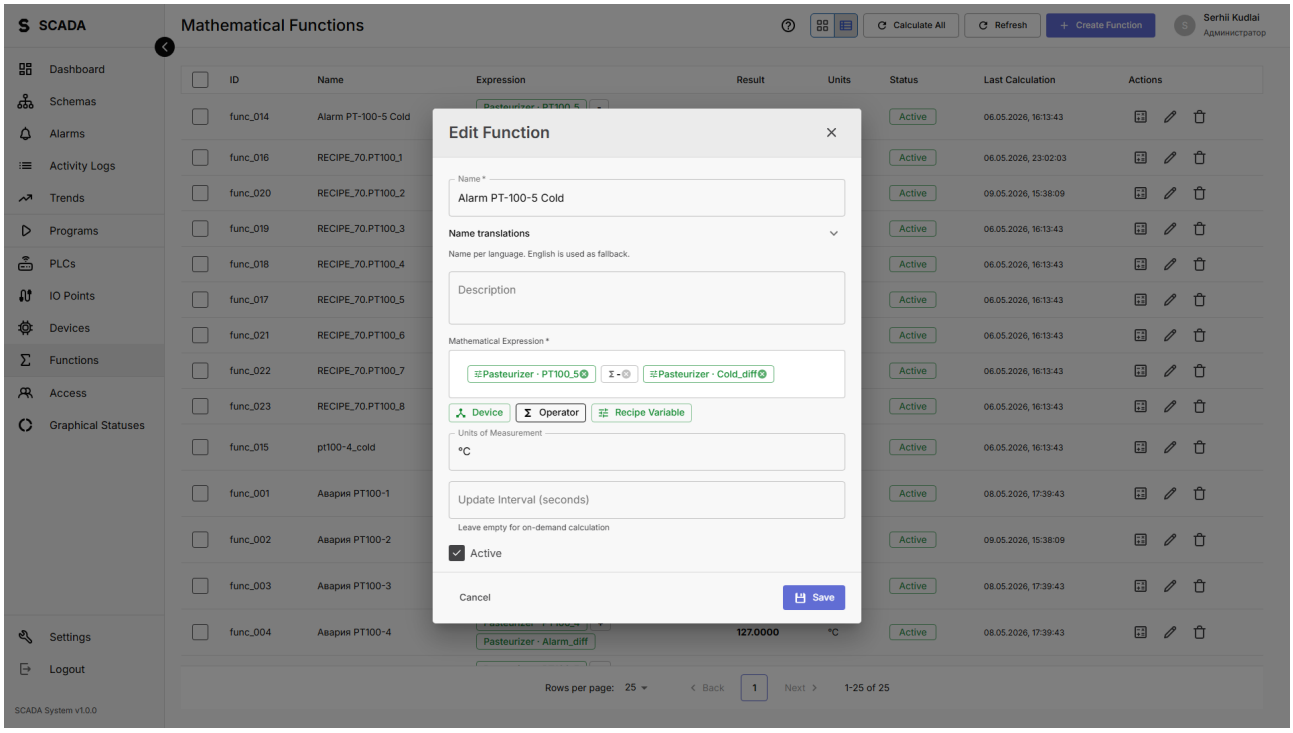
Block type	Description
Number	Constant numeric value
Device	Current device value
Recipe variable	Variable value
Operator	+, −, ×, ÷, %, ^
Parentheses	(and) for grouping
Function	sqrt, log10, round, floor, ceil

Buttons:

Button	Description
Add block	Adds a new block to the expression
Delete block	Removes a block from the expression
Validate	Syntax check
Test	Evaluate with current values
Save	Save the function



Functions list: table with Name, Expression, Result, Unit columns, Edit/Delete/Test buttons.



Expression editor: a row of blocks — [AI_001] [×] [2.5] [+] [sqrt] [(] [AI_002] [)]. Block-add buttons at the bottom. Test result on the right.

Technical documentation

Math functions API:

Method	Endpoint	Description
GET	/api/math-functions/	List functions
POST	/api/math-functions/	Create

Method	Endpoint	Description
PUT	/api/math-functions/{id}/	Update
DELETE	/api/math-functions/{id}/	Delete
POST	/api/math-functions/{id}/calculate/	Calculate
POST	/api/math-functions/{id}/validate_expression/	Validate
POST	/api/math-functions/{id}/test_expression/	Test
POST	/api/math-functions/validate_all/	Validate all

WebSocket: Math function updates

Connection: ws:///ws/math-functions

```
{
  "type": "math_function_update",
  "function_id": 5,
  "name": "Total Flow",
  "expression": "ai_001 * 2.5 + sqrt(ai_002)",
  "result": 157.3,
  "units": "L/h",
  "last_calculated": "2026-04-07T12:00:00Z",
  "calculation_error": null,
  "is_active": true
}
```

18. I/O points

For users

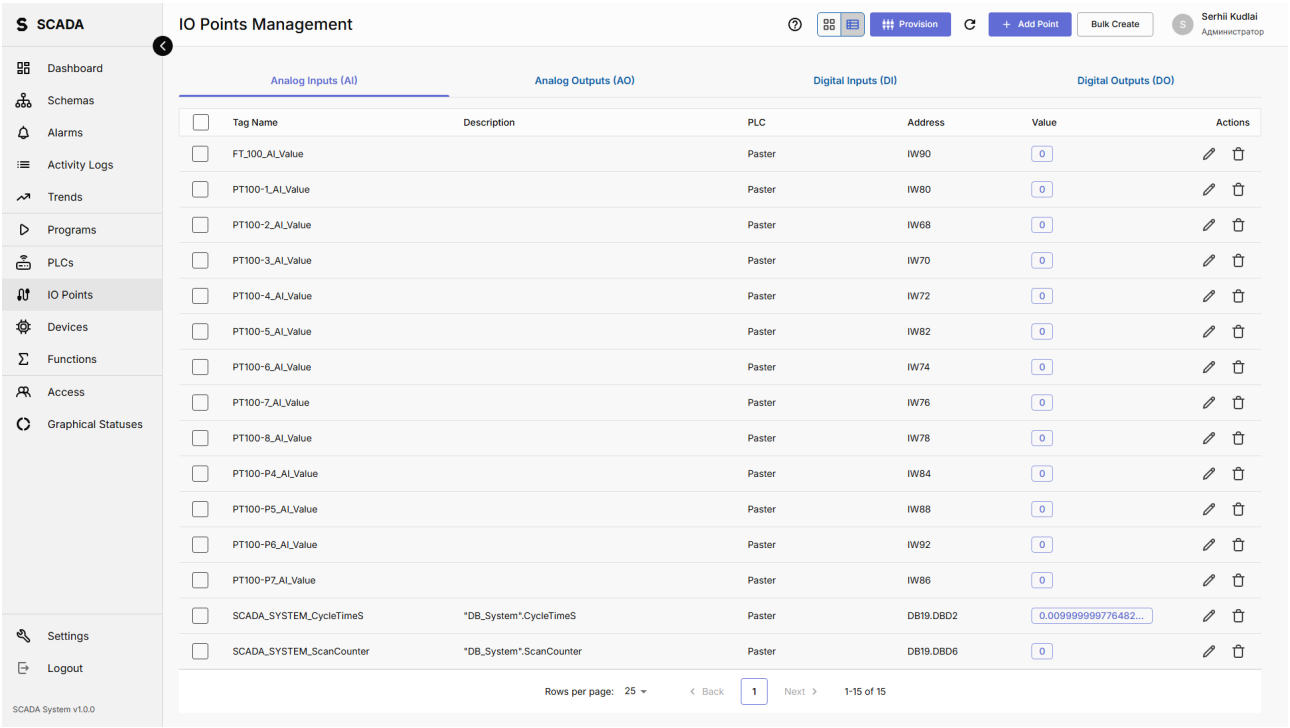
I/O Points define the link between physical PLC signals and logical SCADA devices.

Tabs:

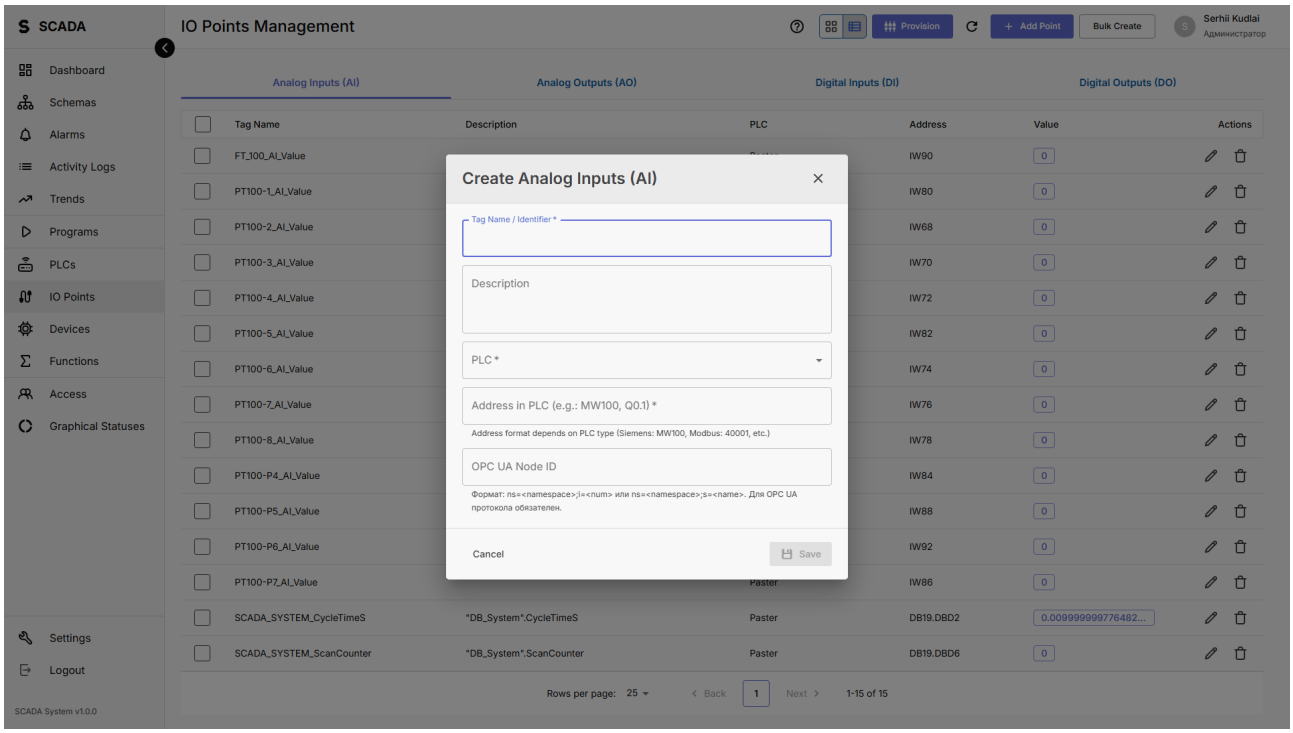
Tab	Description
Analog Inputs	Analog inputs (4–20 mA, 0–10 V)
Analog Outputs	Analog outputs
Digital Inputs	Discrete inputs
Digital Outputs	Discrete outputs

I/O-point form (IoPointDialog):

Field	Description
PLC	Controller selector
Number/Address	PLC address (IW0, QW0, I0.0, Q0.0)
Device	Bind to a logical device
Scaling	Min/Max (for analog)
Data type	INT, REAL, BOOL
Unit	Engineering unit



I/O Points page: 4 tabs (AI, AO, DI, DO), table with PLC, Address, Device, Scale, Unit columns, Add/Edit/Delete buttons.



I/O-point dialog: PLC dropdown, address field, device dropdown, Min/Max Scale fields, Unit.

Technical documentation

I/O points API:

Method	Endpoint	Description
GET	/api/points/analog-inputs/	AI points
POST	/api/points/analog-inputs/	Create AI
GET	/api/points/analog-outputs/	AO points
POST	/api/points/analog-outputs/	Create AO
GET	/api/points/digital-inputs/	DI points
POST	/api/points/digital-inputs/	Create DI
GET	/api/points/digital-outputs/	DO points
POST	/api/points/digital-outputs/	Create DO

For every type: standard CRUD (GET, POST, PUT, PATCH, DELETE) + /validate/ and /test/.

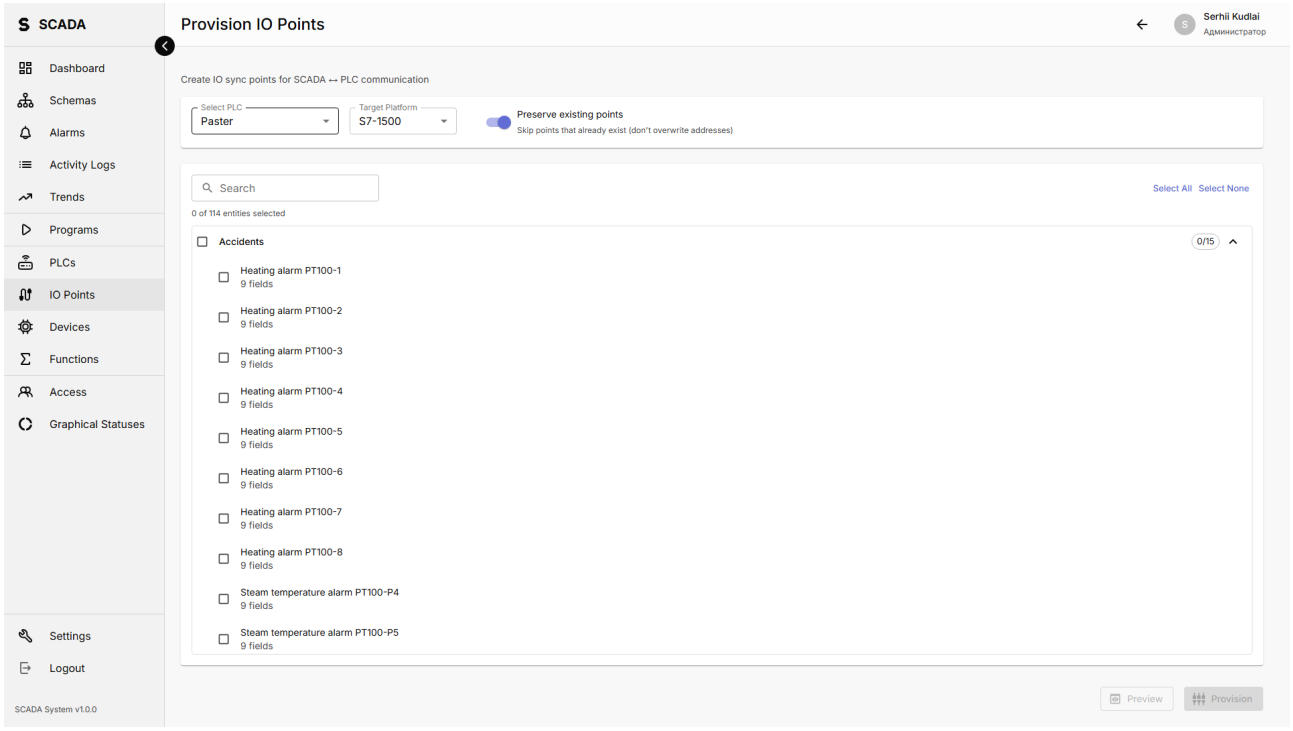
19. I/O provisioning

For users

Bulk creation of I/O points based on device configuration.

Screen elements:

Element	Description
PLC selector	Target controller
PLC type	S7-1200 or S7-1500
Preserve addresses	Checkbox — do not overwrite existing addresses
Entity list	Checkboxes per type (motors, valves, PIDs, AIs, AOs, DIs, DOs, COS, counters, timers, tanks, sequences, events, accidents)
Preview	Result preview
Provision	Run the provisioning



Provisioning page: PLC dropdown, S7-1200/S7-1500 radio, entity list with checkboxes, Preview and Provision buttons.

Technical documentation

Provisioning API:

Method	Endpoint	Description
POST	/api/plc-export/provision-io-points/preview/	Preview
POST	/api/plc-export/provision-io-points/	Execute
GET	/api/plc-export/provision-io-points/entities/	Available entities

20. Graphic statuses

For users

Graphic statuses are icon templates that show device state on mnemonic schemas.

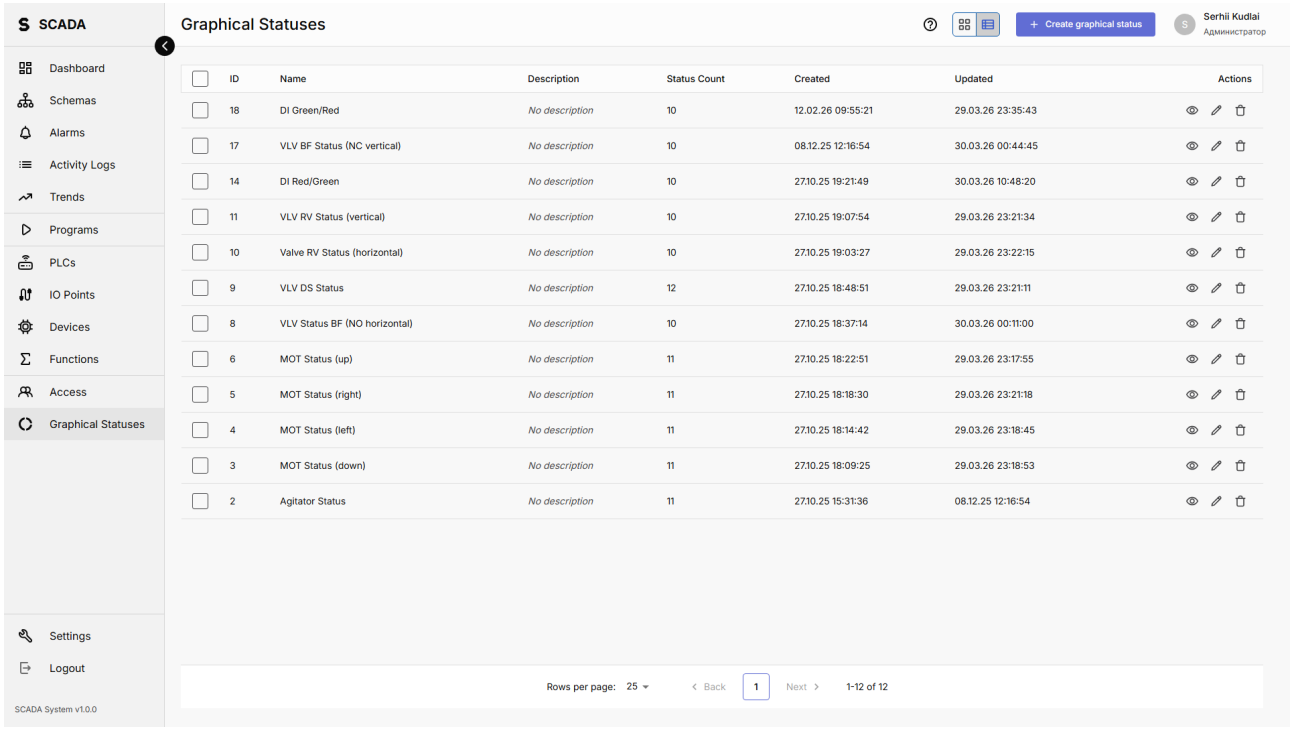
Elements:

Element	Description
Template list	Cards/table
«+» button	Create a template

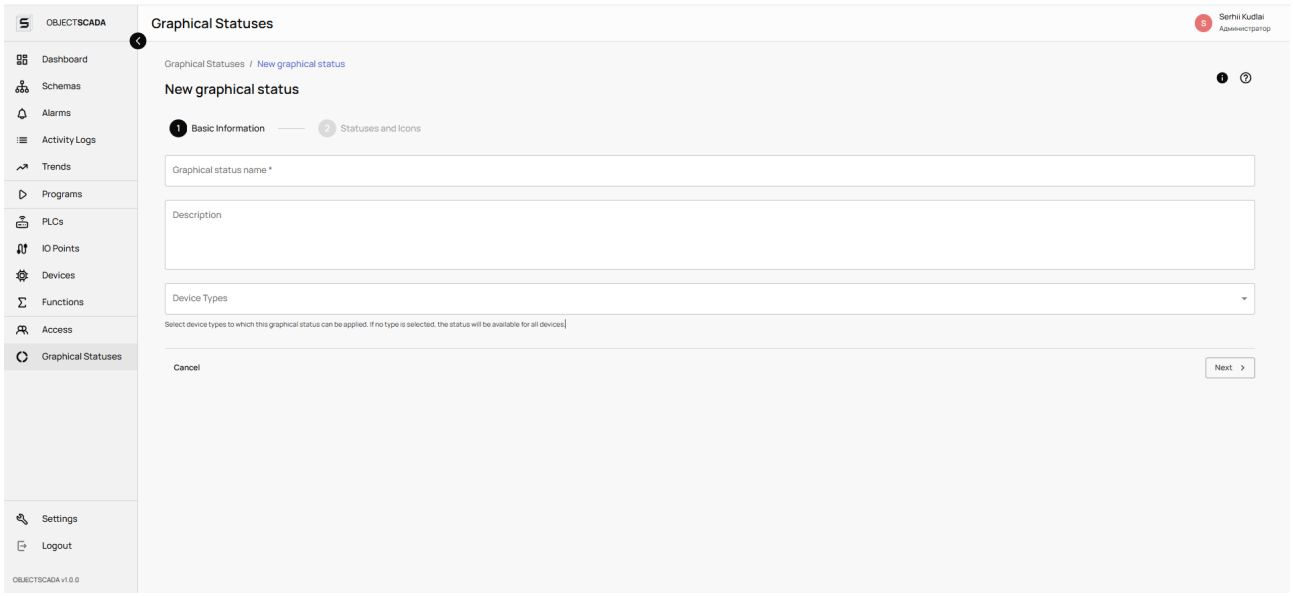
Creation wizard:

- Step 1: Name, description, device type
- Step 2: Statuses and icons
- Status code → icon (URL or upload)
- Gallery of available icons
- Upload custom icons

Button	Description
Add status	New «code → icon» row
Delete status	Remove a row
Upload icon	Upload an icon file
Gallery	Pick from previously uploaded
Previous/Next	Wizard navigation
Save	Save the template



Graphic-status list: cards with icon previews, template name, device type.



Creation wizard: left — list of statuses (code + icon preview), right — icon gallery for selection.

Technical documentation

Graphic statuses API:

Method	Endpoint	Description
GET	/api/graphical-statuses/	List templates
POST	/api/graphical-statuses/	Create
PUT	/api/graphical-statuses/{id}/	Update
DELETE	/api/graphical-statuses/{id}/	Delete
POST	/api/graphical-statuses/upload-icon/	Upload icon (multipart)

Method	Endpoint	Description
GET	/api/graphical-statuses/list-icons/	List icons

21. Hardware configurator

For users

Visual editor of PLC hardware configuration for TIA Portal deployment.

Tabs:

Tab	Description
Rack View	Visual rack editor with modules
PROFINET Topology	Network topology of devices
Address Table	I/O address table

Rack editor:

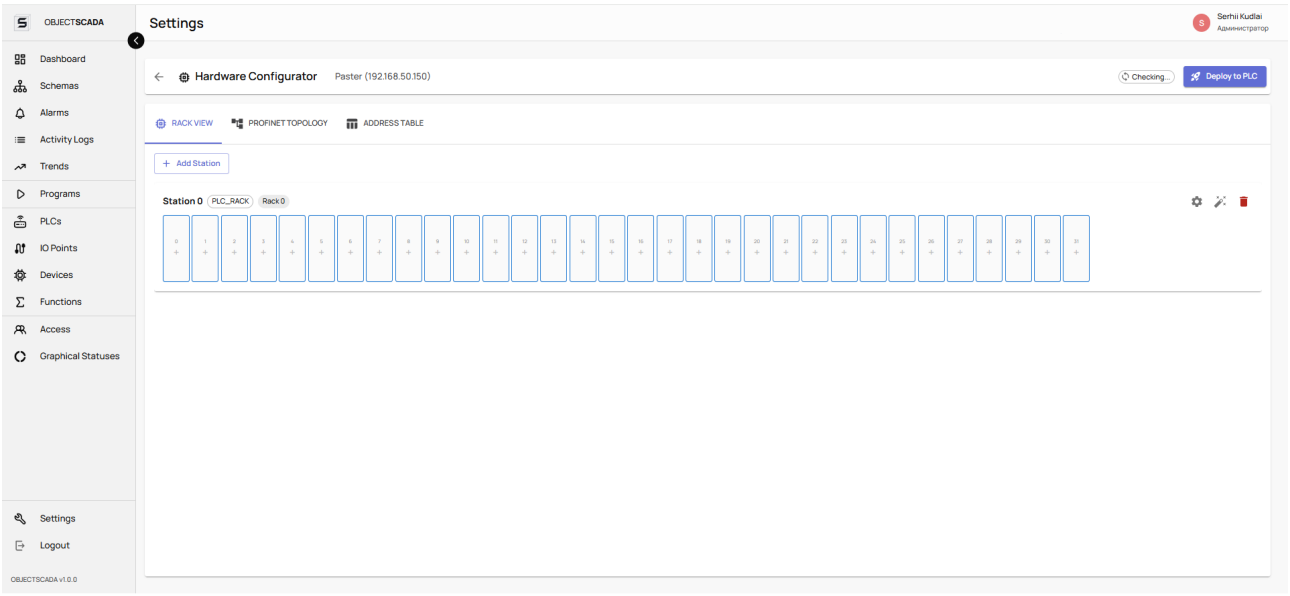
Element	Description
Rack slots	Visual representation of modules in the rack
Module catalogue	Side panel with available modules (drag-drop)
Property panel	Selected-module configuration
Add module	Drag from the catalogue
Remove module	Remove from slot

Toolbar buttons:

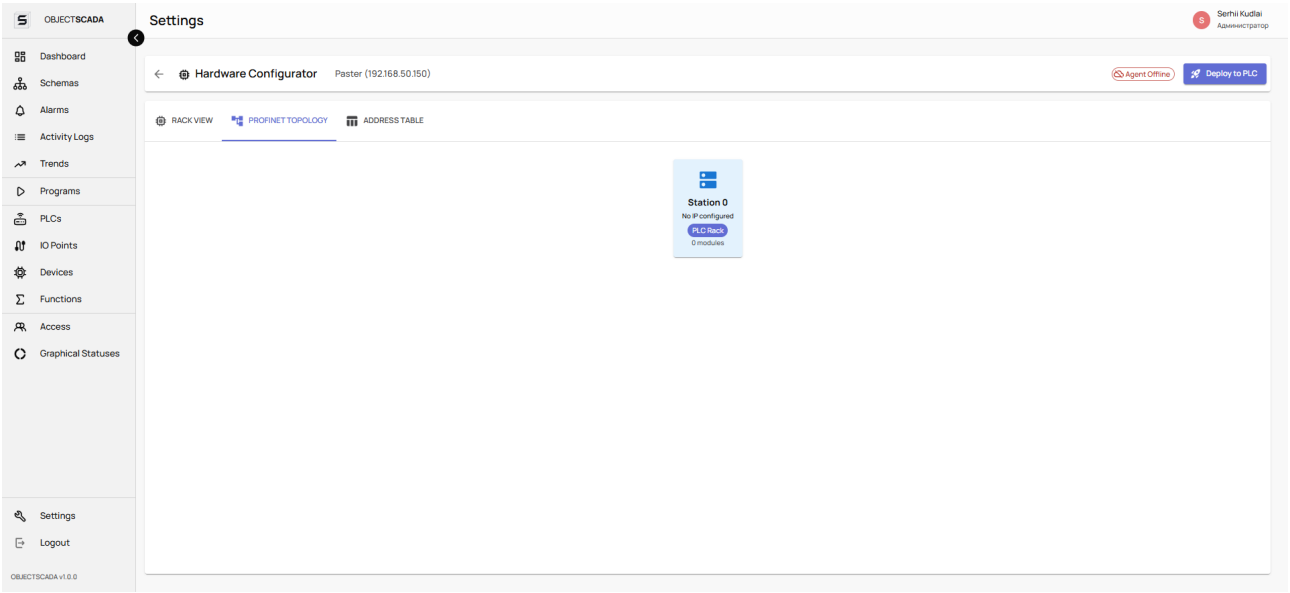
Button	Description
Deploy	Deploy the configuration into TIA Portal
Download	Download the configuration
Import GSD	Import a device descriptor (GSD/GSDML)
TIA Agent Status	TIA Agent connection indicator
Back	Back to the PLC list

Dialogs:

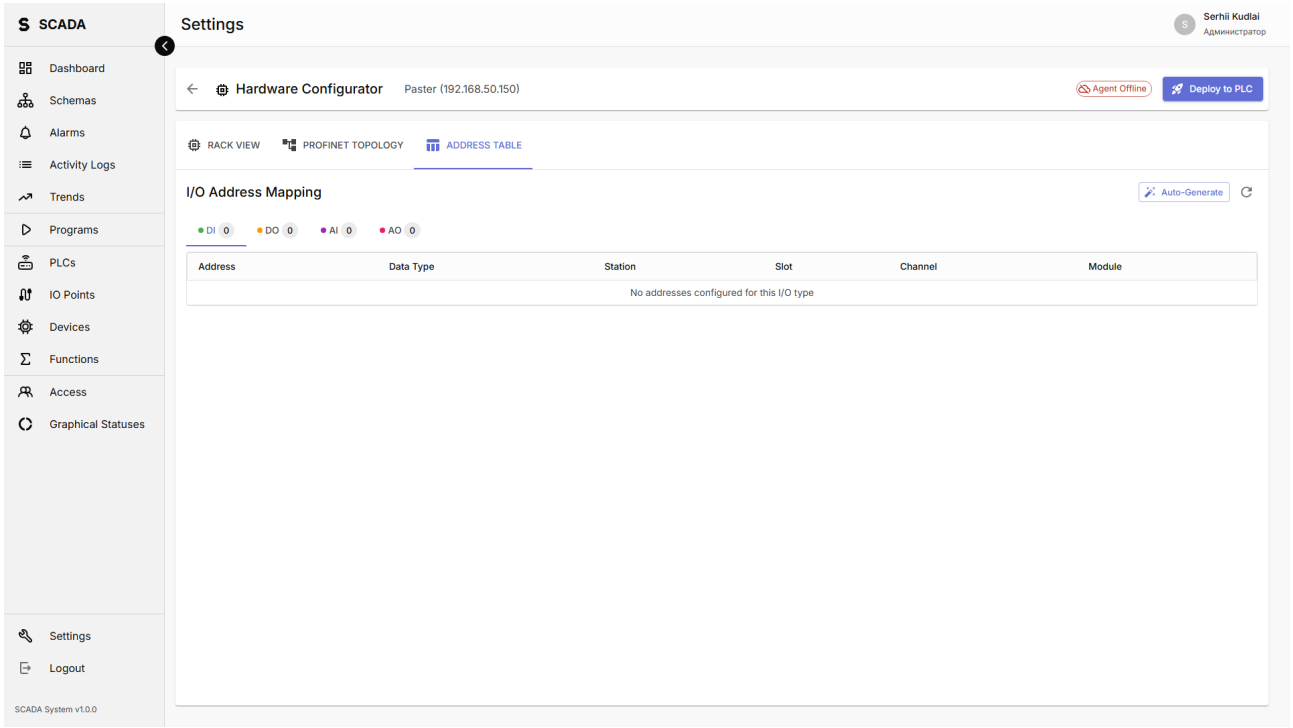
Dialog	Description
TiaDeployDialog	Deployment: target system, progress, status
GsdImportDialog	GSD import: file upload, module preview, confirm
NetworkConfigDialog	Network settings: IP, mask, gateway, PROFINET parameters



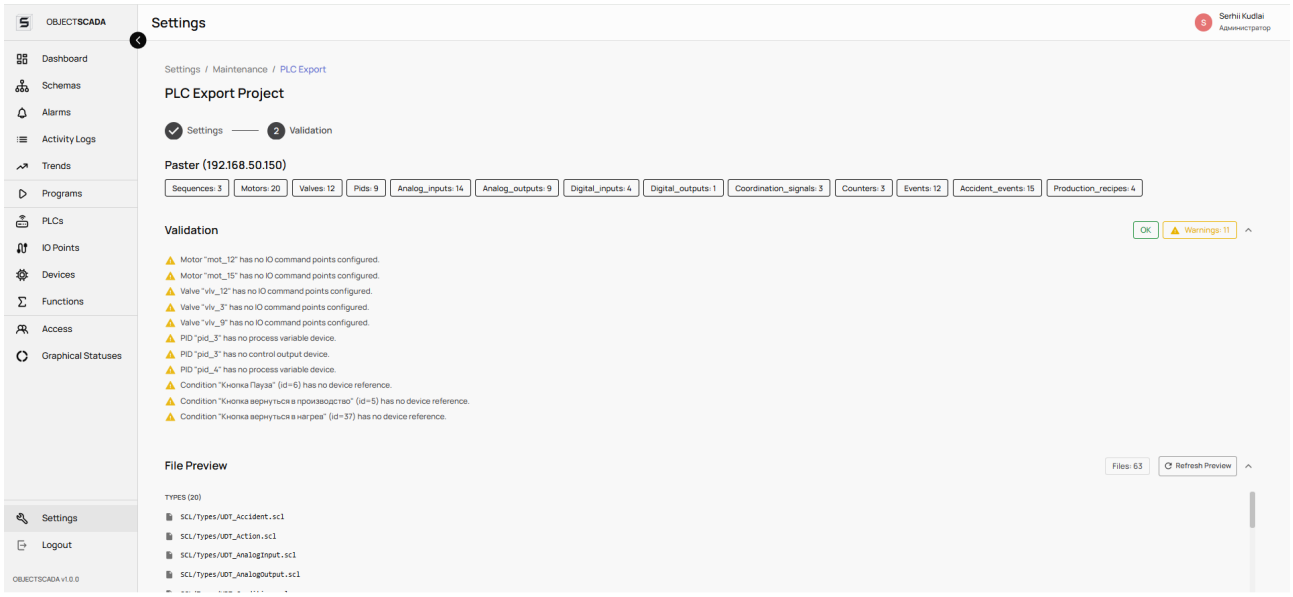
Rack editor: visual rack with modules in slots, module catalogue on the left (slide-out panel), property panel on the right, Deploy button in the toolbar.



PROFINET topology: network diagram with device nodes and links between them.



Address table: Module, Slot, Channel, Address, Type (I/O), Device columns.



Deployment dialog: TIA Agent selector, progress bar, operation log, Deploy/Cancel buttons.

Technical documentation

Configurator API:

Method	Endpoint	Description
GET	/api/hw-catalog/	Module catalogue
POST	/api/hw-catalog/import-gsd/	Import GSD (multipart)
GET	/api/hw-stations/	Stations
POST	/api/hw-stations/	Create a station
POST	/api/hw-stations/{id}/auto-assign-addresses/	Auto-assign addresses

Method	Endpoint	Description
GET	/api/hw-slots/	Slots
POST	/api/hw-slots/	Create a slot
POST	/api/hw-slots/bulk-update/	Bulk update
GET	/api/hw-network/	Network configuration
POST	/api/tia/deploy/	Deploy into TIA Portal
GET	/api/tia-deploy-jobs/{id}/progress/	Deployment progress
GET	/api/tia-agent-config/	TIA Agent configurations
GET	/api/tia-agent-config/{id}/check-connection/	Connection test

22. User management

For users

Create, edit and delete user accounts.

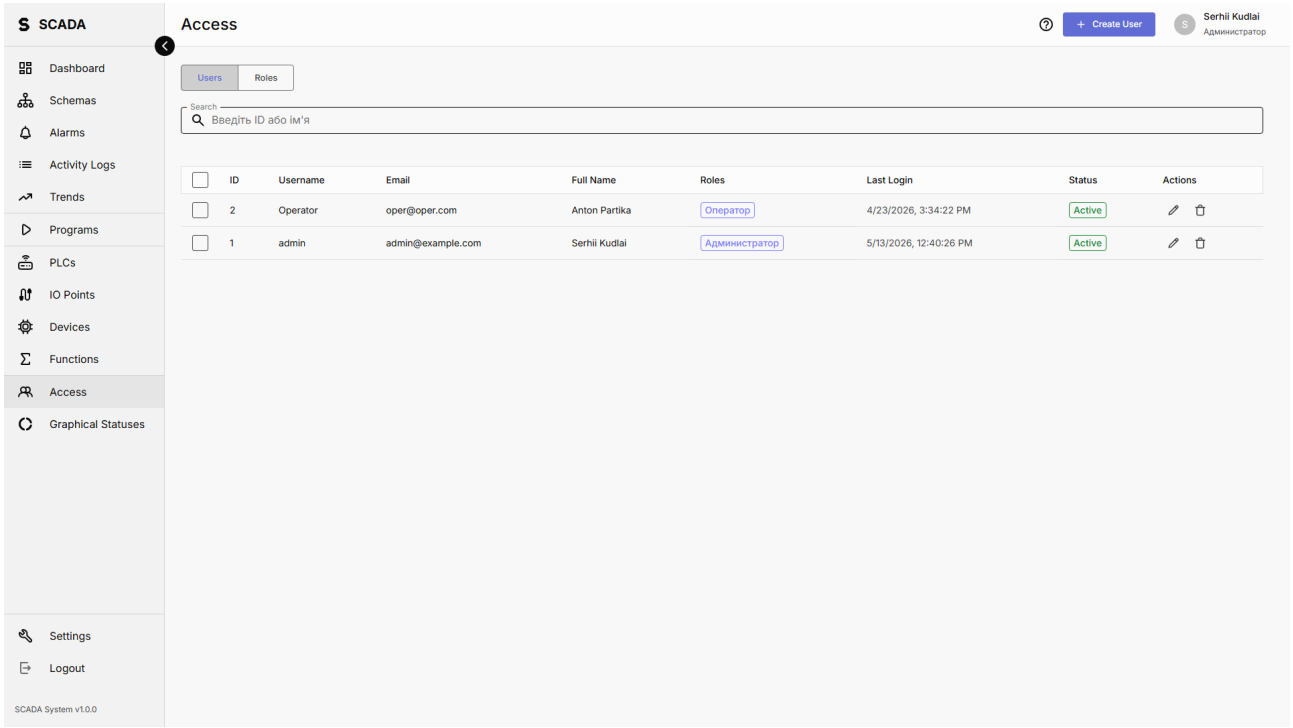
Users table:

Column	Description
Username	Login
Full name	First and last name
Email	Email address
Roles	Assigned roles
Last login	Date and time
Status	Active / Inactive
Actions	Edit / Delete

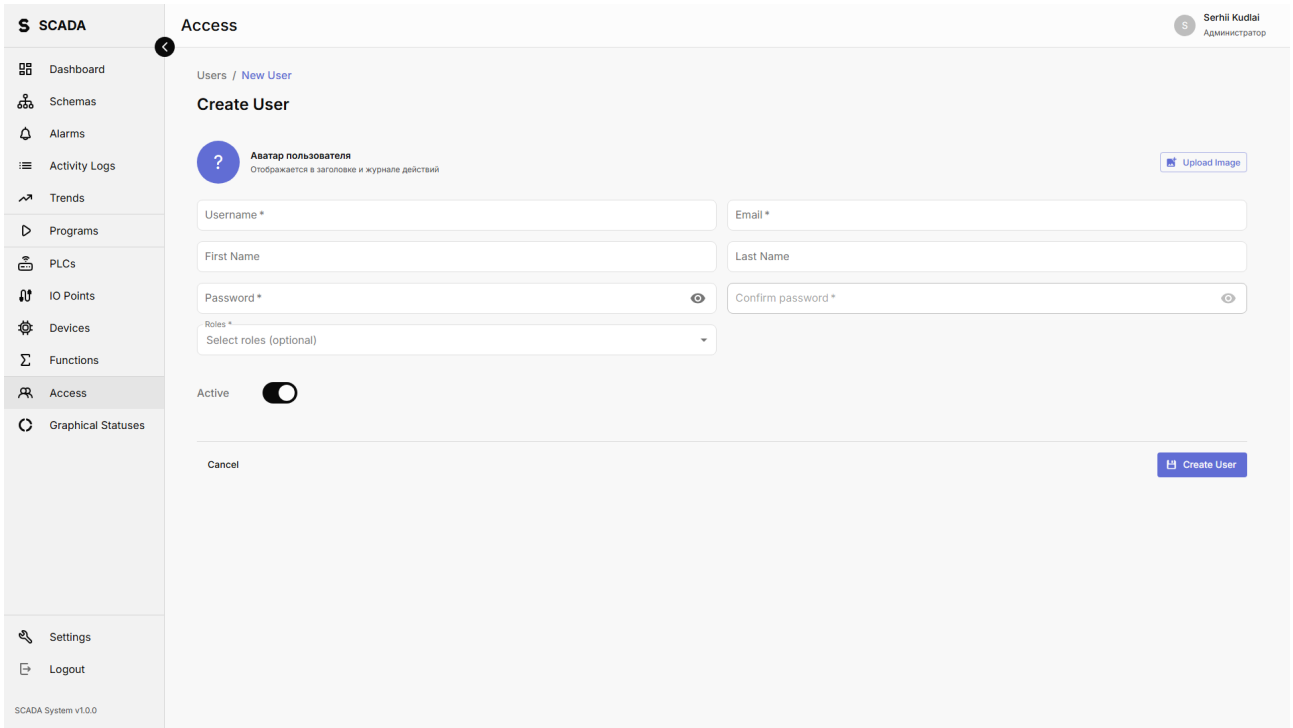
User form:

Field	Description
Username	Unique login
Email	Email address
Password	Password (on create)
Confirm	Password repeat
First / Last name	Real name
Avatar	Profile photo upload
Roles	Role multi-select
Status	Active/Inactive toggle

Button	Description
Save	Save changes
Cancel	Return without saving
Delete	Delete the account (edit mode only)
Change password	Change the user's password



Users table with columns, avatars, role colour badges, status indicator.



Create/edit form: username, email, password, role selector, avatar upload, status toggle.

Technical documentation

Users API:

Method	Endpoint	Description
GET	/api/users/	List users
POST	/api/users/	Create

Method	Endpoint	Description
GET	/api/users/{id}/	Retrieve
PUT	/api/users/{id}/	Update
DELETE	/api/users/{id}/	Delete
GET	/api/users/me/	Current user
POST	/api/users/{id}/set_password/	Change password
POST	/api/users/{id}/upload-avatar/	Upload avatar (multipart)
GET	/api/users/{id}/roles/	User roles
POST	/api/users/{id}/roles/	Assign a role
DELETE	/api/users/{id}/roles/	Revoke a role

23. Role management

For users

Role-based access control. Each role defines a set of permissions.

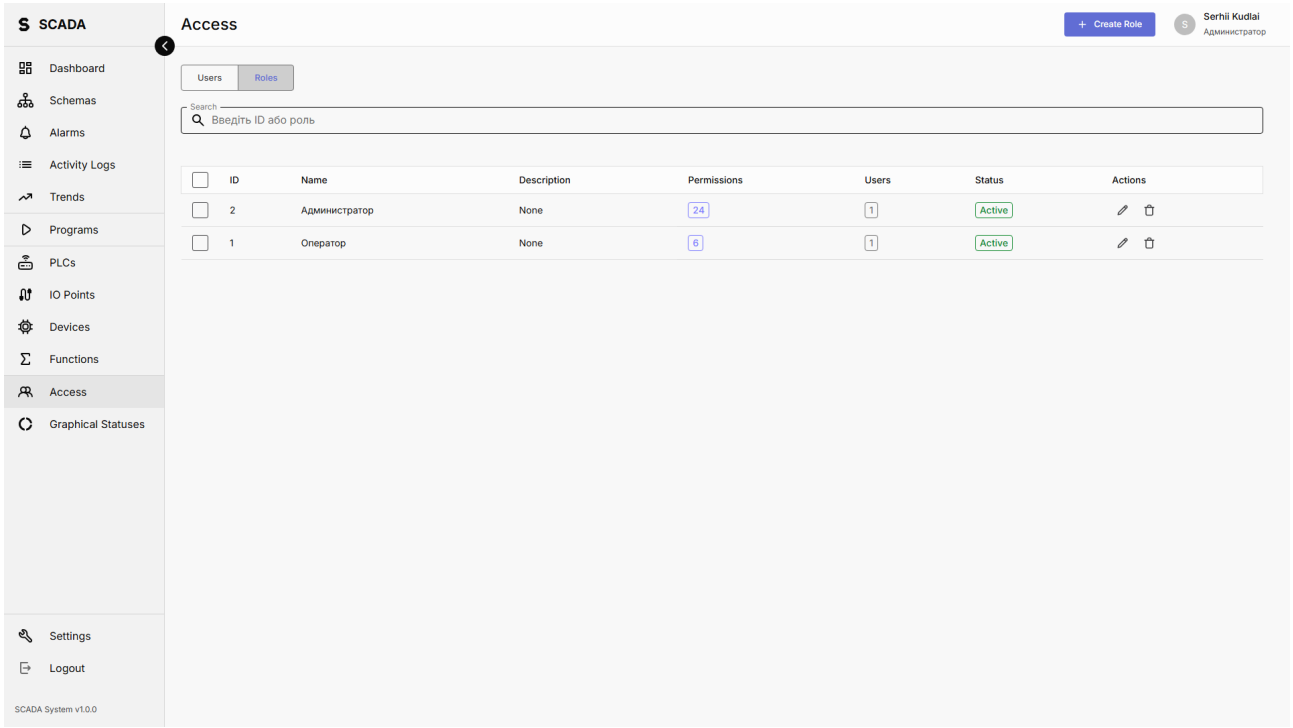
Roles table:

Column	Description
Name	Role name
Description	Text description
Users	Number of users with this role
Actions	Edit / Delete

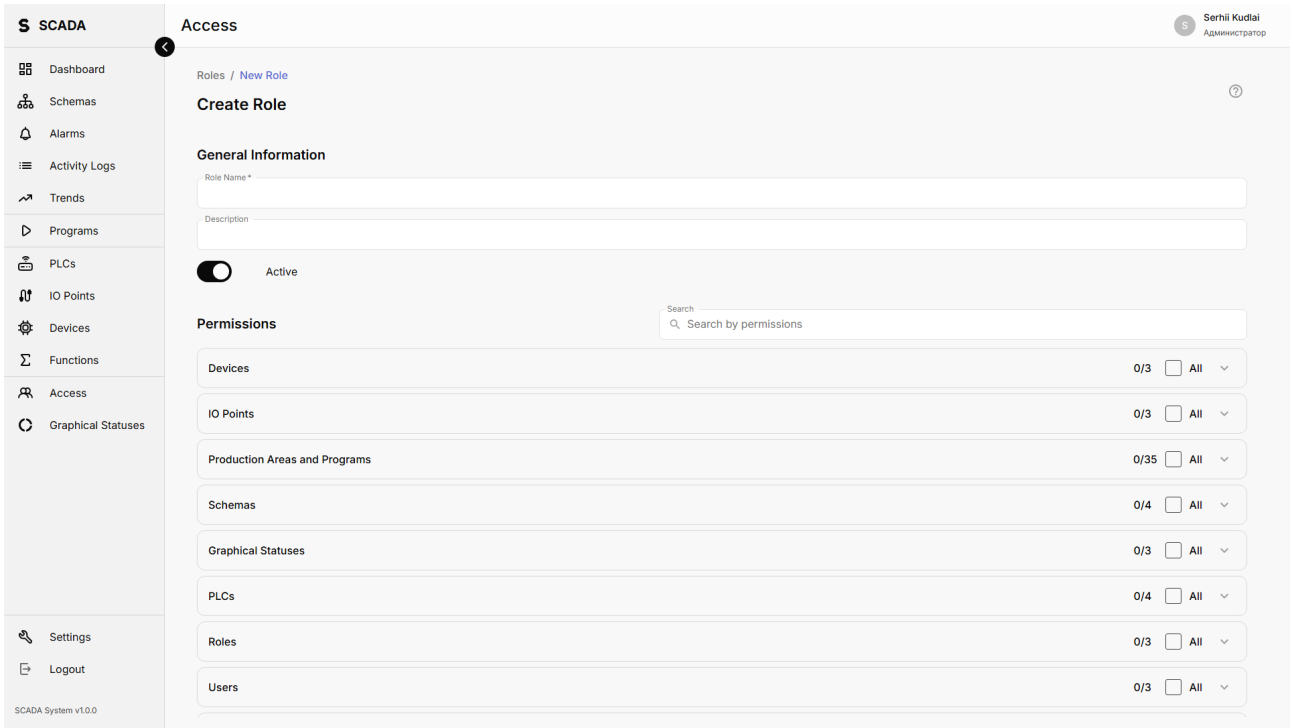
Permission matrix (in the role form):

Category	Permissions
Devices	Create, View, Edit, Delete
PLCs	Create, View, Edit, Delete, Hardware Config
Sequences	Create, View, Edit, Delete, Duplicate
Schemas	Create, View, Edit, Delete
I/O Points	Create, View, Edit, Delete
Users	Create, View, Edit, Delete
Roles	Create, View, Edit, Delete
Accidents	Acknowledge, Resolve, Delete
Trends	View, Export, Delete data
Graphic statuses	Create, View, Edit, Delete
Management	Connections, License, Updates

Button	Description
Select All	Select every permission
Deselect All	Clear every permission
Save	Save the role



Roles list: table with name, description, user count, Edit/Delete buttons.



Role form: name, description, checkbox matrix by category (Devices, PLCs, Sequences, Schemas...). Select All/Deselect All buttons.

Technical documentation

Roles API:

Method	Endpoint	Description
GET	/api/roles/	List roles
POST	/api/roles/	Create

Method	Endpoint	Description
GET	/api/roles/{id}/	Retrieve
PATCH	/api/roles/{id}/	Update
DELETE	/api/roles/{id}/	Delete
GET	/api/roles/{id}/permissions/	Role permissions
GET	/api/permissions/categories/	Permission categories
GET	/api/permissions/actions/	Permission actions

24. Activity log

For users

Audit of all user actions in the system.

Screen elements:

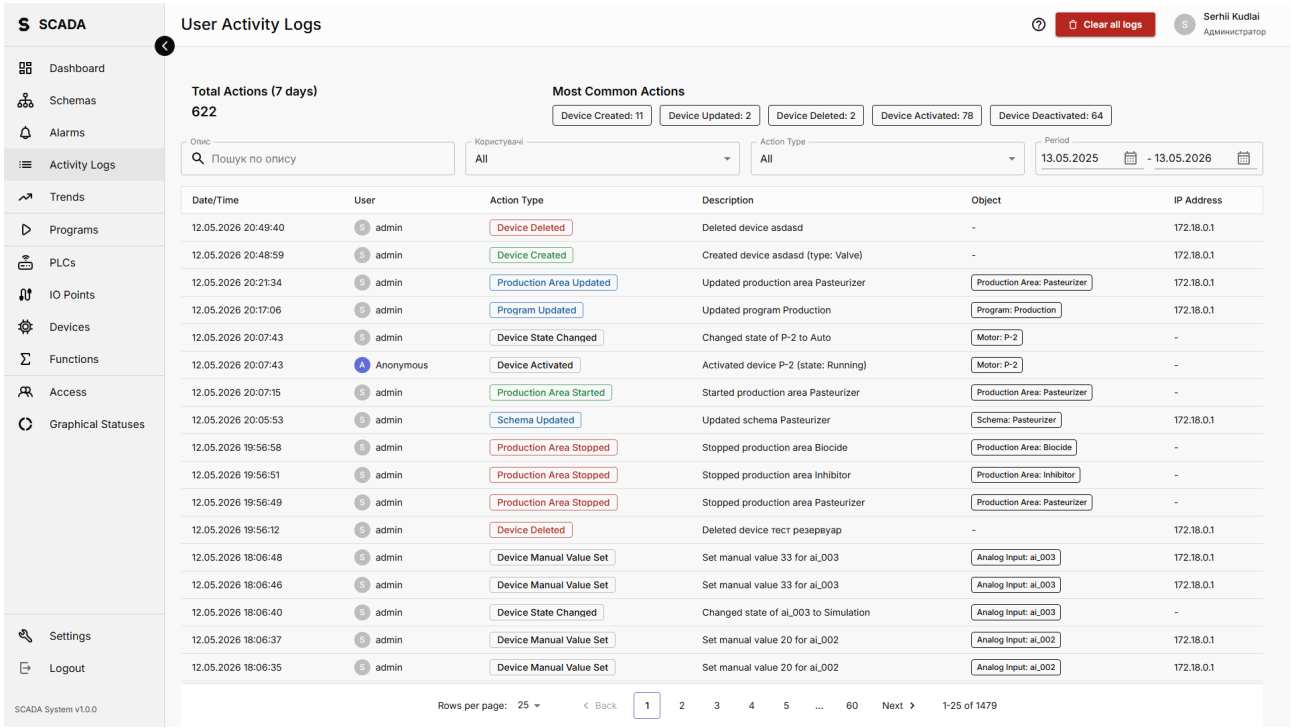
Column	Description
Time	Action date and time
User	Who performed the action
Action	Operation type (create, update, delete, login...)
Description	Action details
IP address	Source IP

Filters:

Filter	Description
By user	User selector
By action	Operation type
By date	Date range
Search	Text search

Statistics:

- Total number of actions
- Summary by user
- Summary by action type



Activity log: table with filters on top, statistics on the right (charts by action type and user). Pagination at the bottom.

Technical documentation

Activity log API:

Method	Endpoint	Description	Parameters
GET	/api/user-activity-logs/	List of logs	?user_id=X&action_type=X&limit=100&offset=0
GET	/api/user-activity-logs/recent/	Last 24 h	—
GET	/api/user-activity-logs/statistics/	Statistics	—
DELETE	/api/user-activity-logs/clear/	Clear (admin)	—

WebSocket: Live log

Connection: ws:///ws/activity-log

```
{
  "type": "activity_log_entry",
  "id": 1234,
  "action_type": "device_update",
  "description": "Updated motor MOT_001 mode to manual",
  "user": {"id": 1, "username": "admin"},
  "content_object": {"type": "motor", "id": "mot_001"},
  "metadata": {},
  "created_at": "2026-04-07T12:00:00Z"
}
```

25. Settings

For users

The Settings page has multiple tabs:

General tab

Parameter	Description
Theme	Colour scheme selector (Blue, Green, Orange, Pink, Purple, Red)
Dark mode	Light/Dark toggle
Language	Interface language selector (EN, UK, RU, DE, FR, PL, AR...)
Design version	V1 / V2
AI assistant	Enable/disable the AI assistant

Notifications tab

Parameter	Description
Email notifications	On/off
Push notifications	On/off
SMS notifications	On/off
Severity filter	Minimum severity for notifications

Connections tab

Element	Description
TIA Agent list	TIA Portal connection configurations
Add	New TIA Agent (host, port)
Test	Test connection
Status indicator	Green/red

License tab

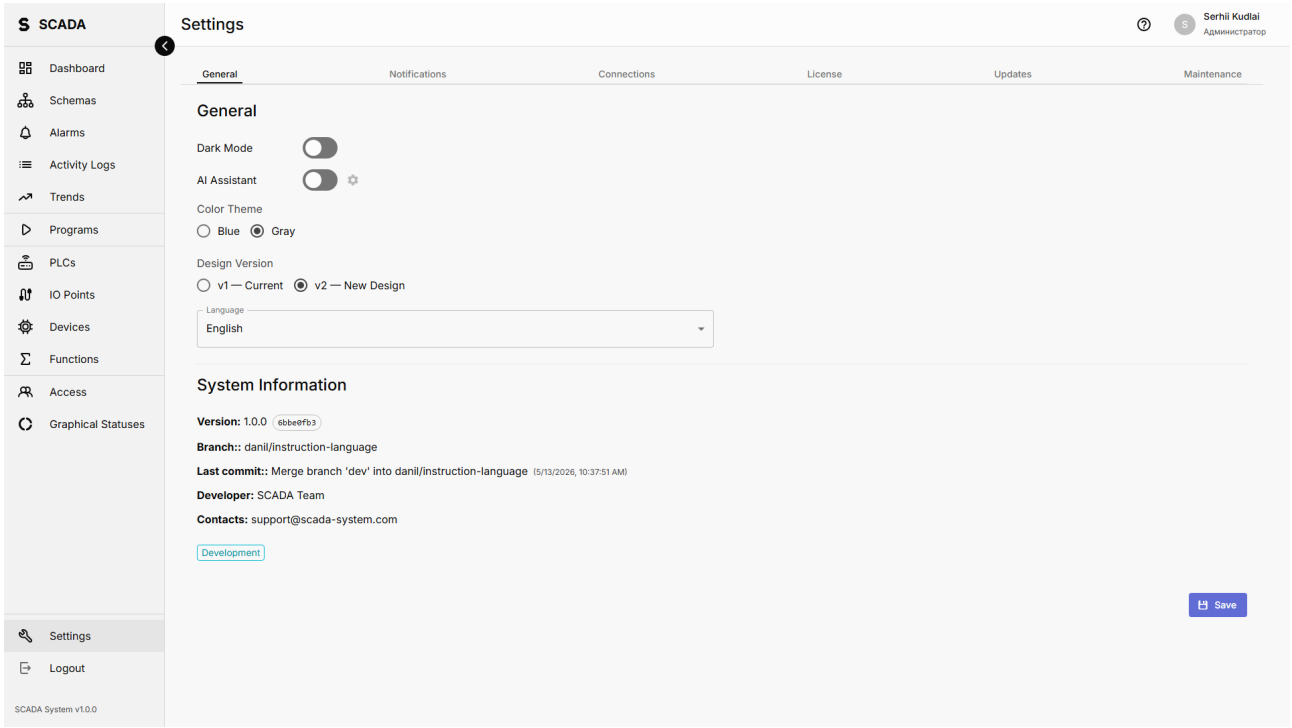
Field	Description
Status	Valid / Invalid / Trial
Type	Lifetime / Trial / Subscription
Days remaining	Countdown
Company	Company name
Expiry date	License expiry date
Installation ID	Unique installation identifier

Updates tab

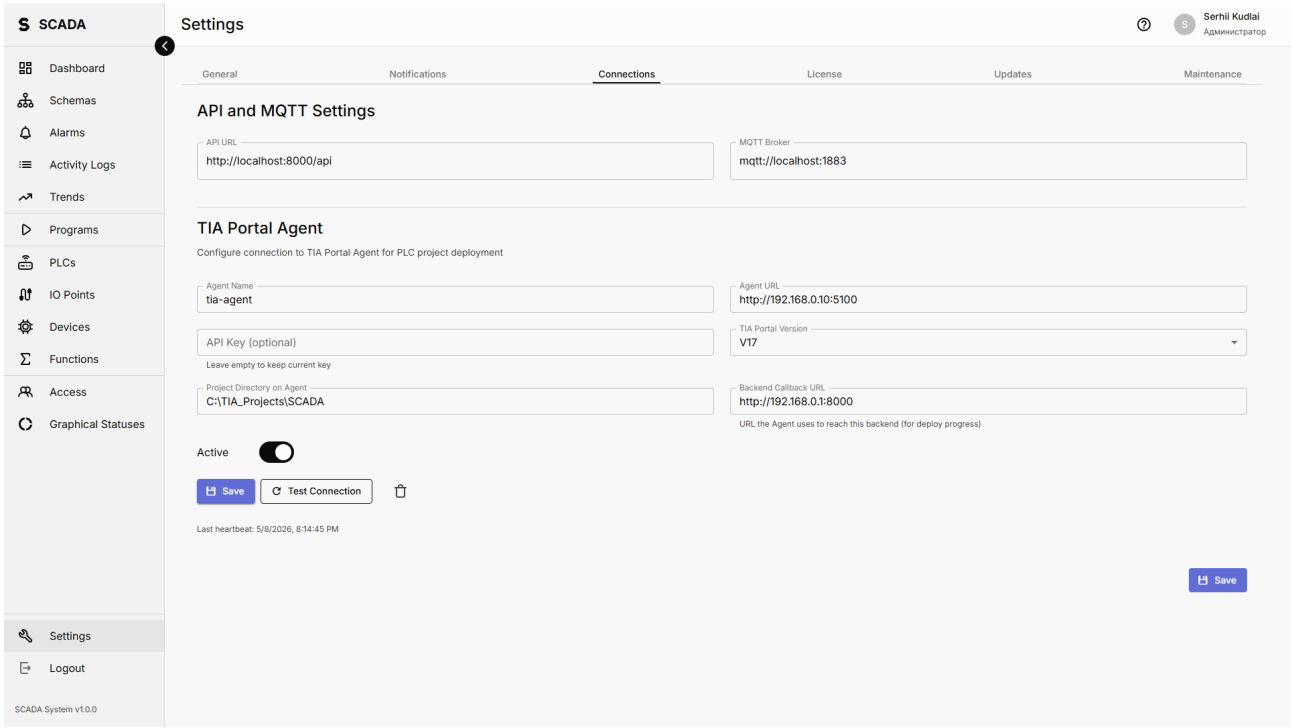
Element	Description
Current version	Current version number
Check for updates	Check button
Download update	Download button
Progress	Update progress indicator
Auto-update	Toggle

Maintenance tab

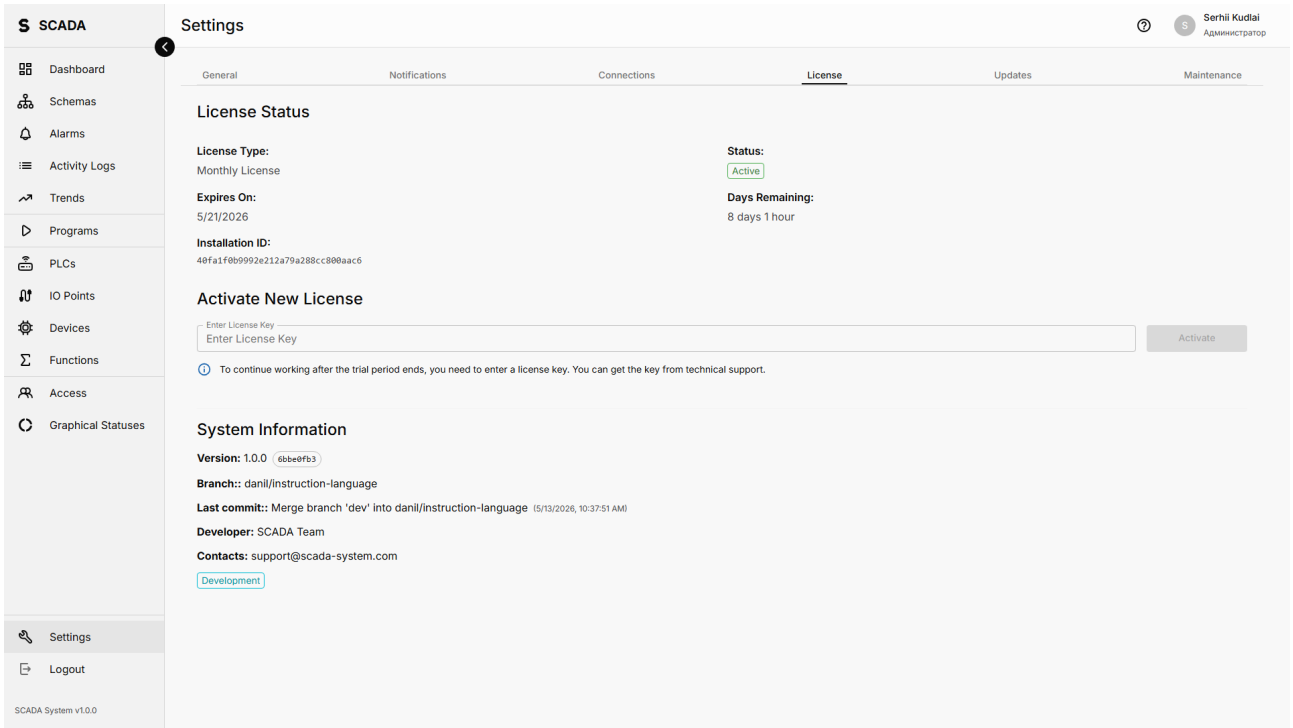
Element	Description
Database	DB maintenance operations
Cache	Clear cache
Logs	Log management
System health	Health checks
Backup/Restore	Backup and restore
Disk cleanup	Remove temporary files
Smoke test	Runtime health check



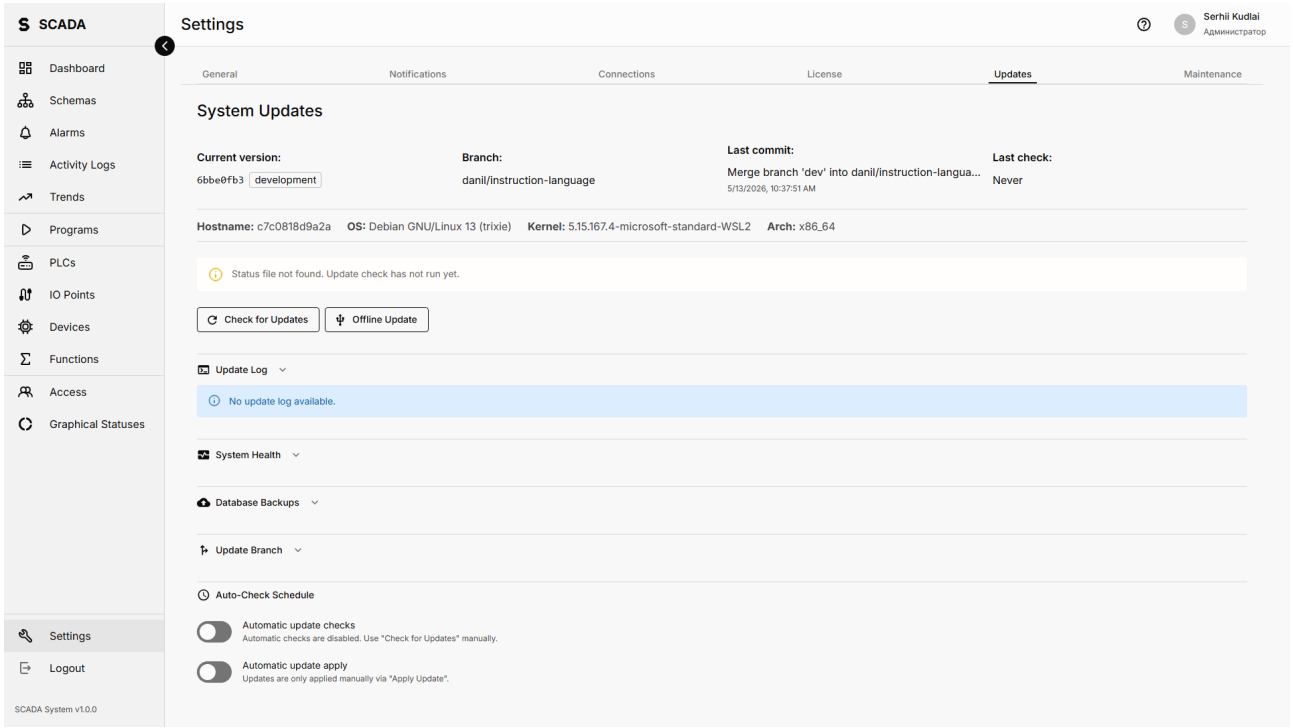
General tab: Dark Mode toggle, Color Picker (6 colours), language dropdown, Design V1/V2 toggle, AI Assistant toggle.



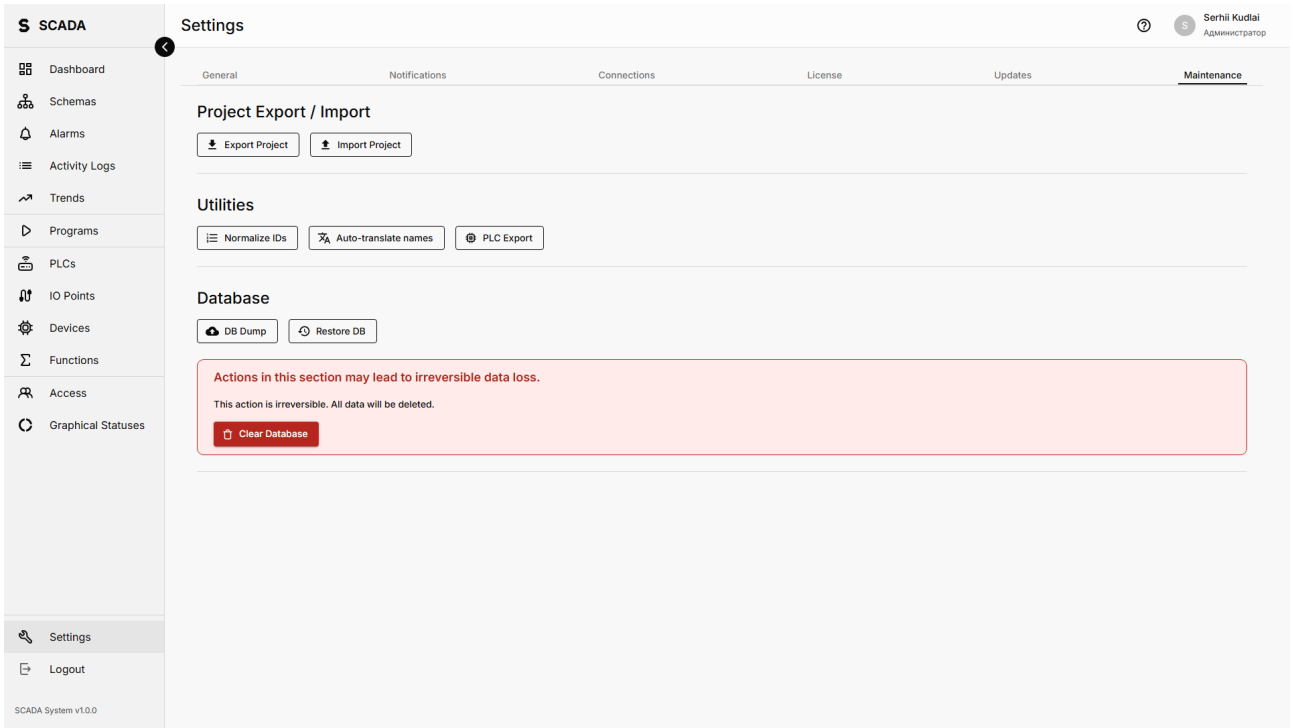
Connections tab: TIA Agent list with host/port, Add/Edit/Delete/Test buttons, status indicators.



License tab: license status (green badge), type, days remaining, company, expiry date.



Updates tab: current version, Check for Updates button, progress bar, update log.



Maintenance tab: Backup DB, Restore DB, Clear Cache, Disk Cleanup, Smoke Test buttons with descriptions.

Technical documentation

User settings API:

Method	Endpoint	Description
GET	/api/user/language/	Get language
POST	/api/user/language/	Set language

Method	Endpoint	Description
GET	/api/user/dark-mode/	Get dark mode
POST	/api/user/dark-mode/	Set dark mode

License API:

Method	Endpoint	Description
GET	/api/license/status/	License status
POST	/api/license/activate/	Activate
POST	/api/license/generate/	Generate a key

Updates API:

Method	Endpoint	Description
GET	/api/system/update-status/	Update status
POST	/api/system/update-check/	Check for updates
POST	/api/system/update-apply/	Apply update
GET	/api/system/update-progress/	Progress
POST	/api/system/update-cancel/	Cancel
POST	/api/system/update-rollback/	Rollback
GET	/api/system/update-history/	Update history
GET	/api/system/update-backups/	Backups
POST	/api/system/update-backups/{filename}/restore/	Restore from backup
POST	/api/system/disk-cleanup/	Disk cleanup
GET	/api/system/smoke-test/	Smoke test
GET	/api/system/diagnostics/	Diagnostics download

WebSocket: Update notifications

```
{
  "type": "system_update",
  "event": "update_progress",
  "progress_percent": 45,
  "current_step": "Applying migrations...",
  "status": "in_progress"
}
```

26. AI assistant

For users

A built-in AI assistant that lets you manage SCADA through natural language.

Elements:

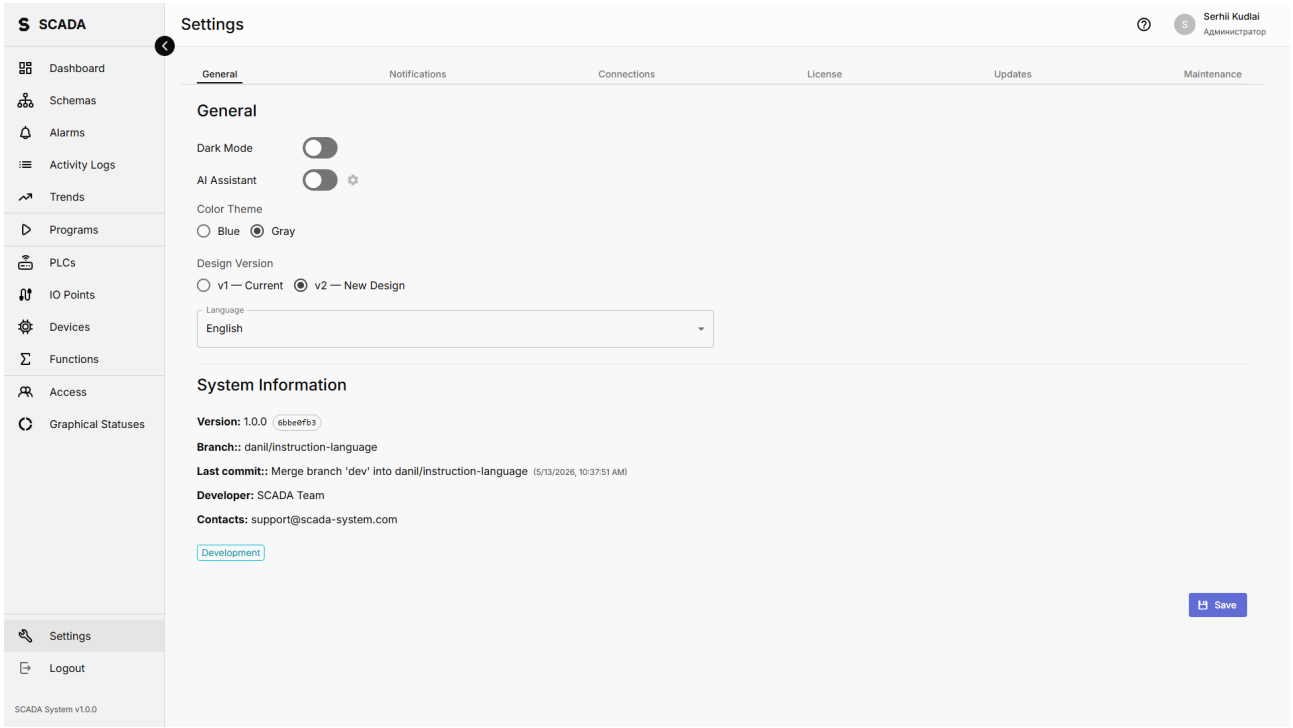
Element	Description
AI button (FAB)	Floating button in the bottom-right corner
Chat panel	Slide-out/draggable panel
Session list	Side panel with the conversation history
Input field	Text field for messages
Send button	Sends a message
Attach button	Upload an image (P&ID drawing)
Settings	AI configuration (model, API key)

AI capabilities:

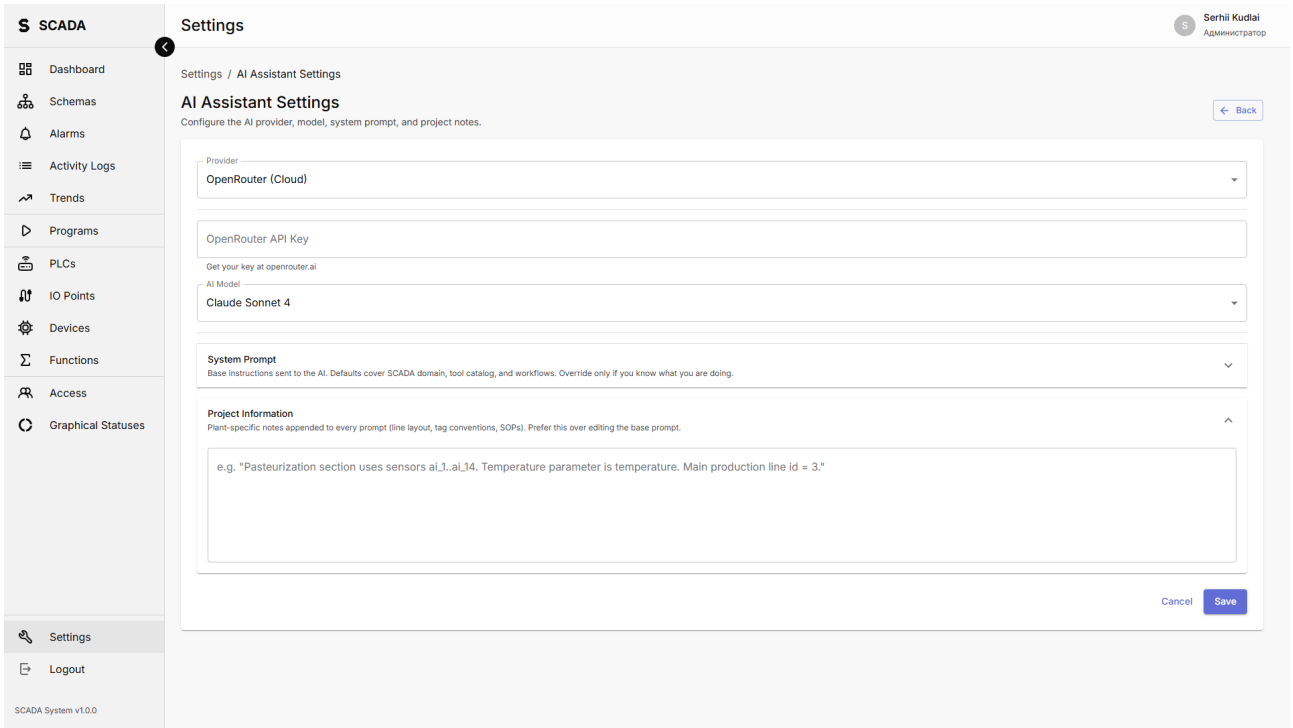
Function	Description
Device control	«Turn on motor MOT_001»
Information queries	«Which devices are currently in alarm?»
Element creation	«Create a new analog input AI_010»
P&ID analysis	Upload image → element recognition → placement suggestions
Help	Q&A about the system

AI settings (AiSettingsDialog):

Field	Description
API key	OpenRouter key (encrypted with AES/Fernet)
Model	LLM model selector
Temperature	Response creativity (0–1)
Max tokens	Maximum response length



Floating AI button in the bottom-right corner: round, with a robot/chat icon.



AI chat panel: left — session list, right — conversation (user and AI messages), bottom — input with Send and Attach buttons.

Technical documentation

AI API (FastAPI microservice, BaseURL: `/ai-api/v1`):

Method	Endpoint	Description
POST	/sessions	Create a session
GET	/sessions	List sessions

Method	Endpoint	Description
DELETE	/sessions/{id}	Delete a session
PATCH	/sessions/{id}	Rename
GET	/sessions/{id}/messages	Session messages
POST	/chat/{id}/messages	Send a message (SSE stream)
POST	/images/analyze	Analyze an image (multipart)
POST	/images/apply	Apply a proposal
GET	/settings	Get settings
PATCH	/settings	Update settings
GET	/health	Health check

SSE events (with stream=true):

Event	Description
token	A token of the response text (streaming)
tool_call	AI calls a tool (devicetools, schematools, query_tools)
tool_result	Tool call result
error	Error
done	Response finished

SSE format:

```
event: token
data: {"content": "Motor "}

event: token
data: {"content": "MOT_001 "}

event: tool_call
data: {"name": "get_device_status", "arguments": {"device_id": "mot_001"}}

event: tool_result
data: {"result": {"status": "running", "value": 1}}

event: token
data: {"content": "is running now."}

event: done
data: {}
```

27. PLC project export

For users

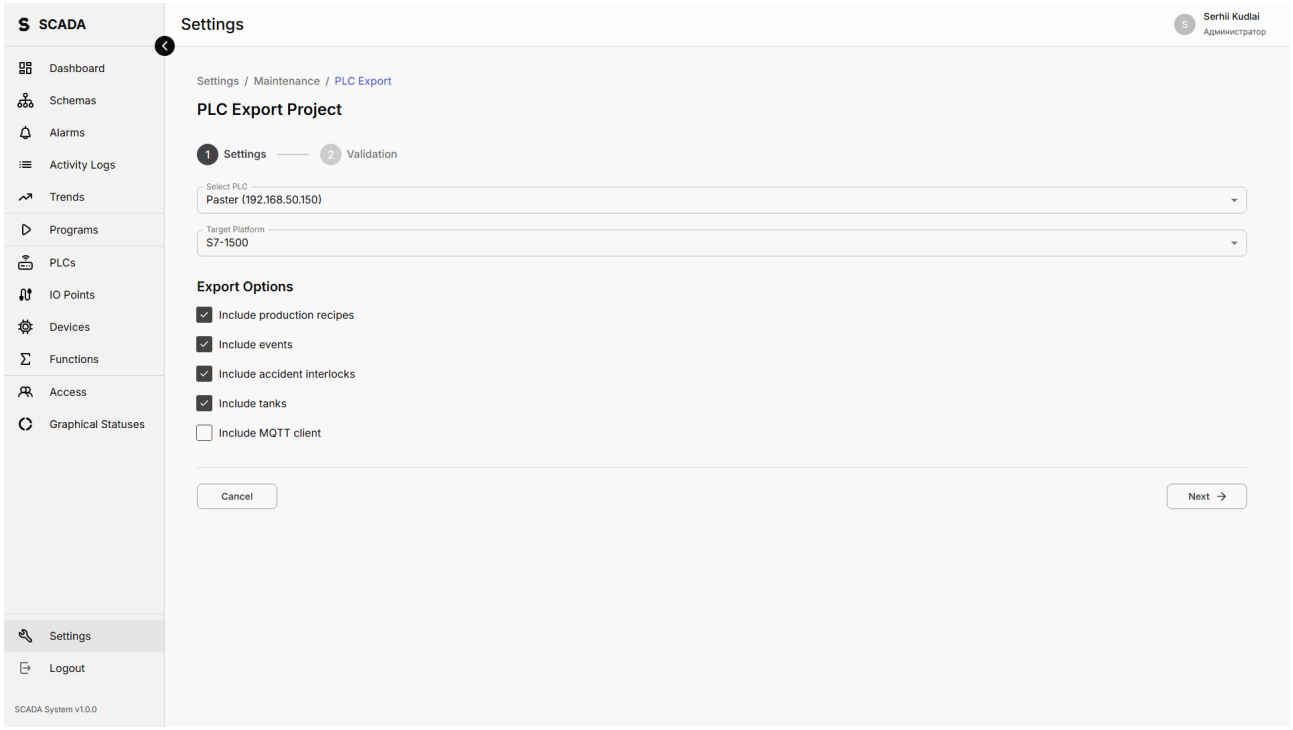
Generate a Siemens TIA Portal project from the SCADA configuration.

Export dialog (PlcExportDialog):

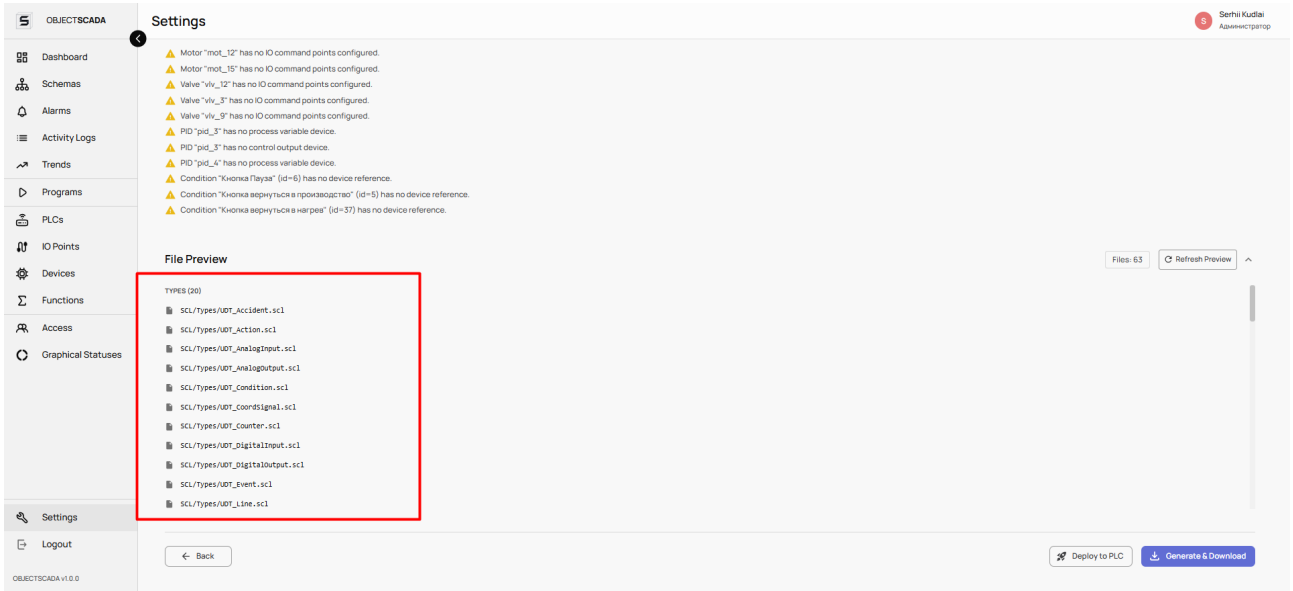
Element	Description
PLC selector	Target controller
PLC type	S7-1200 / S7-1500
Include recipes	Checkbox — add recipes
Include events	Checkbox — add events
Include accidents	Checkbox — add accidents
Include tanks	Checkbox — add tanks
Include MQTT	Checkbox — add MQTT communication
MQTT Broker IP	MQTT broker address
MQTT Topic Prefix	Topic prefix
Preview	SCL code preview
Generate	Download project ZIP
Deploy to TIA	Deploy via TIA Agent

Generated project contents:

- SCL code of function blocks (Motor FB, Valve FB, PID FB, AI FB, AO FB, DI FB, DO FB, COS FB, Counter FB, Timer FB, Tank FB)
- Sequence FB (state machine)
- Event/Accident FBs
- Data Blocks (Instance DBs, Recipe DBs)
- OB1 Main (calls every FB)
- XML Tag Tables (SimaticML)



Export dialog: PLC dropdown, S7-1200/S7-1500 radio, Include checkboxes (Recipes, Events, Accidents, Tanks, MQTT), MQTT config fields, Preview/Generate/Deploy buttons.



SCL code preview: left — file tree (FBs, DBs, Tag Tables), right — SCL file contents with syntax highlighting.

Technical documentation

PLC Export API:

Method	Endpoint	Description
GET	/api/plc-export/preview/	Code preview
POST	/api/plc-export/validate/	Validation
POST	/api/plc-export/generate/	Generate ZIP
GET	/api/plc-export/address-map/	Address map

28. Project export and import

For users

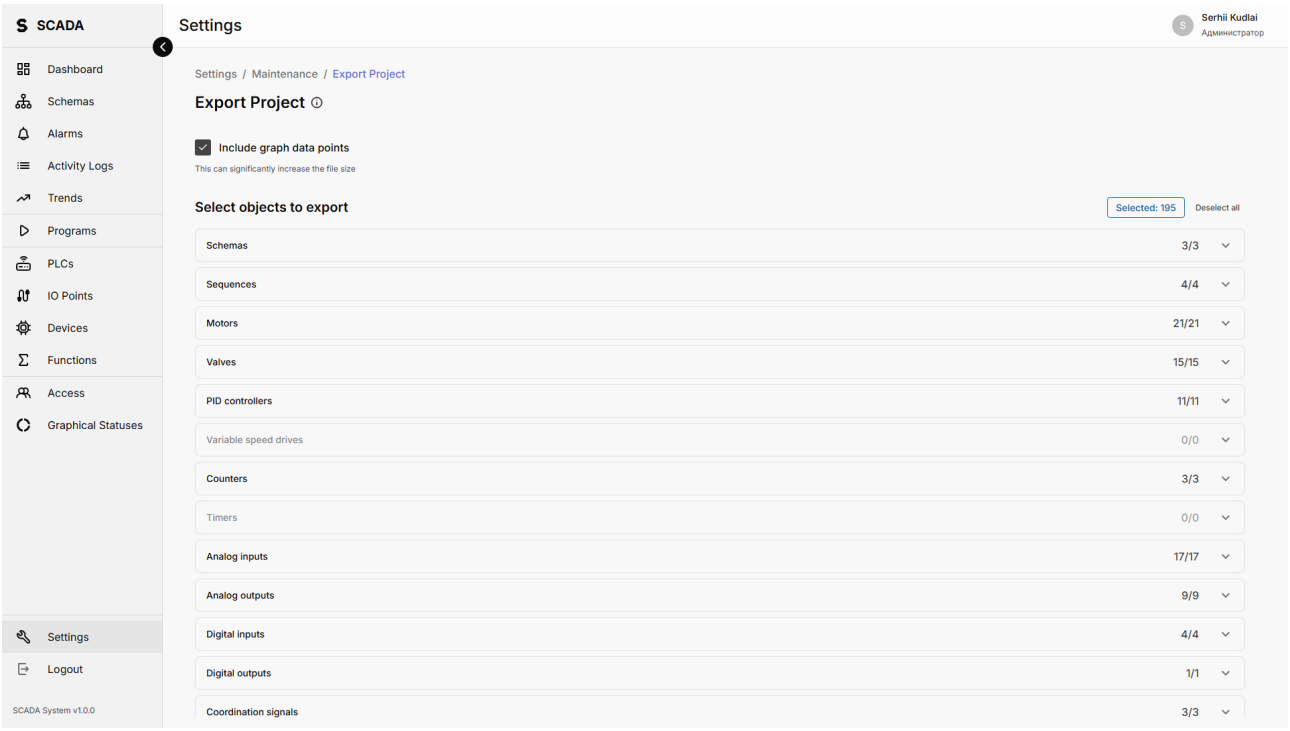
Full SCADA project export/import for migration between installations.

Export dialog (ExportProjectDialog):

Element	Description
Object checkboxes	Pick types to export: Devices, Schemas, Sequences, Recipes, Events, Accidents, Options, Trends, Alarms, Users, Roles
Select All	Select all
Include graphs	Export trend data
Search	Object filter
Export	Download export file
Progress	Progress indicator

Buttons:

Button	Description
Export project	Start export of selected objects
Import project	Upload file → preview → apply
DB backup	Create a database backup
Restore DB	Restore from backup



Export dialog: list of types with checkboxes, Select All button, «Include Graphs» toggle, Export button, progress bar.

Technical documentation

Project API:

Method	Endpoint	Description
GET	/api/project/export-models/	List exportable models
POST	/api/project/export/	Export
POST	/api/project/import/preview/	Import preview
POST	/api/project/import/	Import
POST	/api/project/backup-db/	DB backup
POST	/api/project/restore-db/	Restore
POST	/api/project/normalize-ids/	Normalize IDs
POST	/api/project/clear-database/	Clear DB
POST	/api/project/export-graph-data/	Export charts
POST	/api/project/import-graph-data/	Import charts
GET	/api/project/graph-data-stats/	Graph statistics

29. Tanks

For users

Tanks are vessels with automatic volume calculation based on sensor data.

Tank parameters:

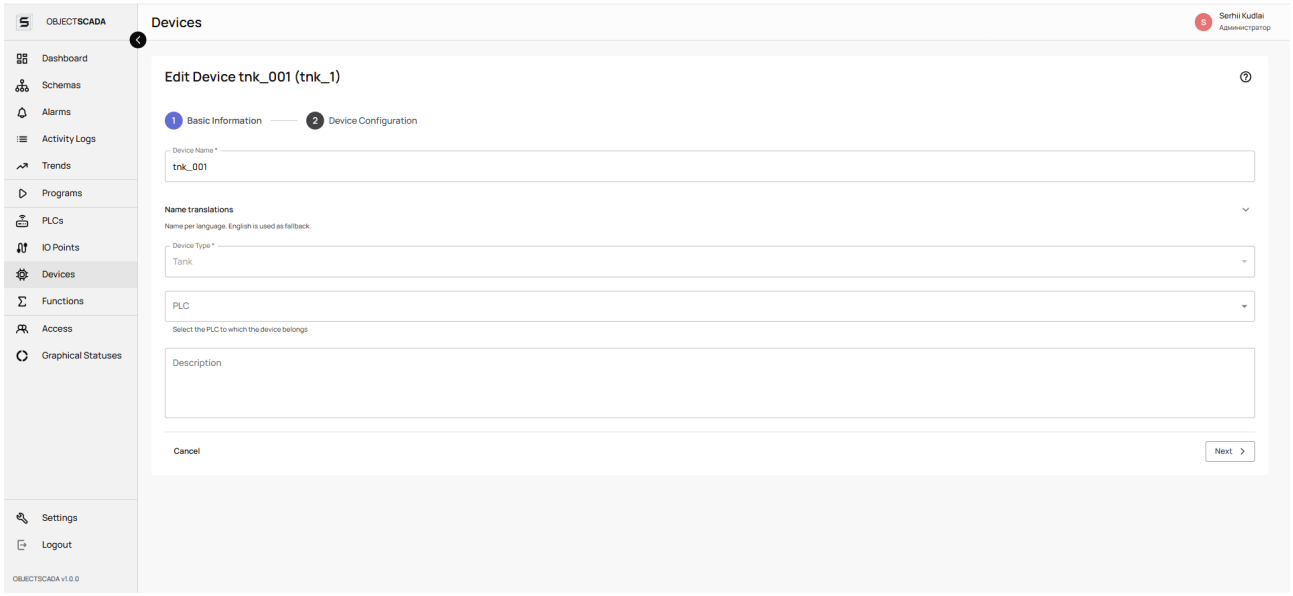
Field	Description
Level sensor	Bind to an AI device (level transducer)
Top pressure sensor	Top pressure (optional)
Bottom pressure sensor	Bottom pressure (for ΔP)
Alarm levels	HH, H, L, LL — alarm thresholds

Volume calculation methods:

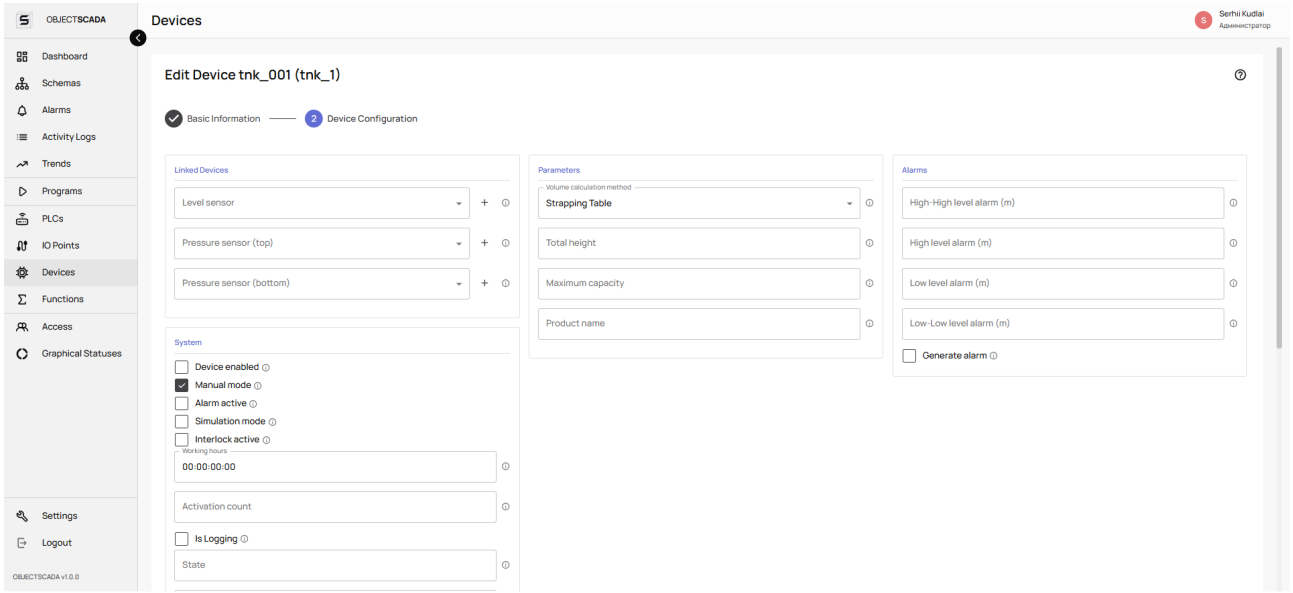
Method	Description
Strapping Table	Calibration table (interpolation sensor_value → volume)
Geometric	Geometric (sections: cylinder, cone, dome, dish)

Tank constructor on the schema:

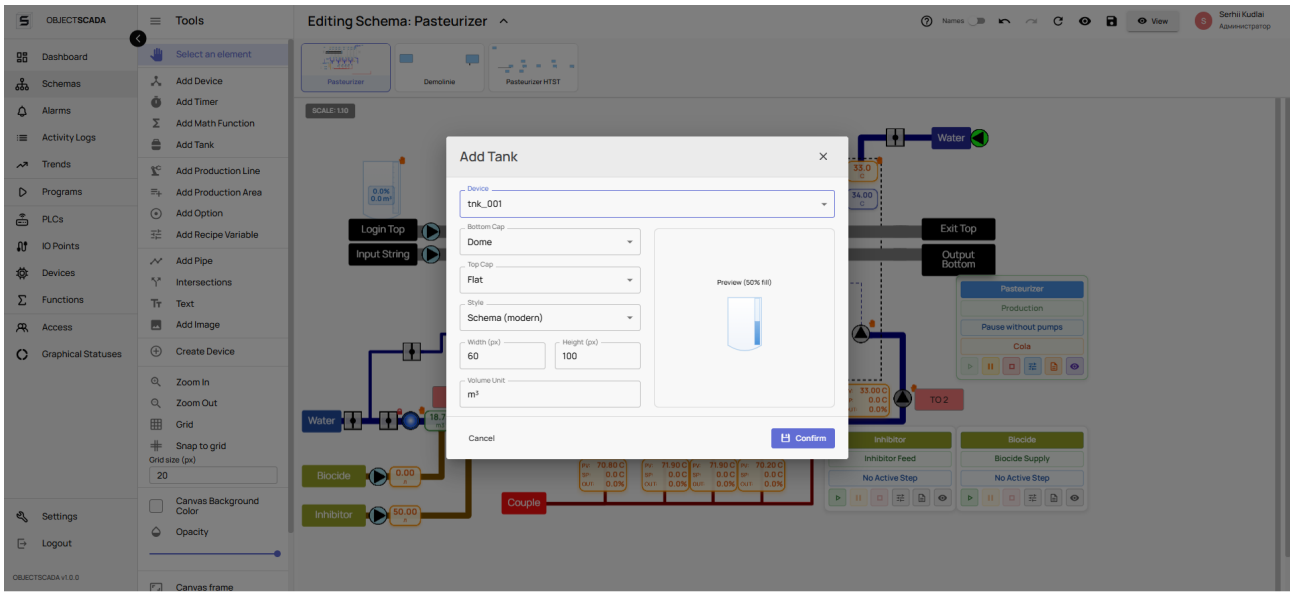
Element	Description
Device selector	Dropdown of existing TNK devices
Bottom cap type	FLAT / DOME / DISH / CONE
Top cap type	FLAT / DOME / DISH
Width / Height	Visual dimensions
SVG preview	Preview of the tank shape
Auto-place sensors	Sensors are placed along the right wall



Tank form: sensor binding (Level Sensor dropdown, Pressure Top, Pressure Bottom), alarm thresholds (HH, H, L, LL — numeric fields).



Strapping table: two columns — Sensor Value and Volume, Add Row, Delete Row, Import CSV, Export CSV buttons.



Tank constructor: left — settings (device, caps, dimensions), right — SVG preview of the selected shape.

Technical documentation

Tanks API:

Method	Endpoint	Description
GET	/api/tanks/	List tanks
POST	/api/tanks/	Create
PUT	/api/tanks/{id}/	Update
DELET	/api/tanks/{id}/	Delete
POST	/api/tanks/{id}/calculate_volume/	Recalculate volume
POST	/api/tanks/{id}/import_strapping_table/	Import table (CSV)
GET	/api/tanks/{id}/export_strapping_table/	Export table
GET	/api/tanks/{id}/volume_at_level/	Volume at level (?sensor_value=X)
GET	/api/tank-sections/	Sections (?tank=ID)
GET	/api/tank-strapping-entries/	Table rows (?tank=ID)

Rust Worker — periodic calculation (every 1000 ms):

File scada-worker/src/tasks/tank_volume.rs — polls level and pressure sensors, computes volume from the strapping table or geometry, writes the result back to the DB and publishes it through Redis.

Application navigation (sidebar)

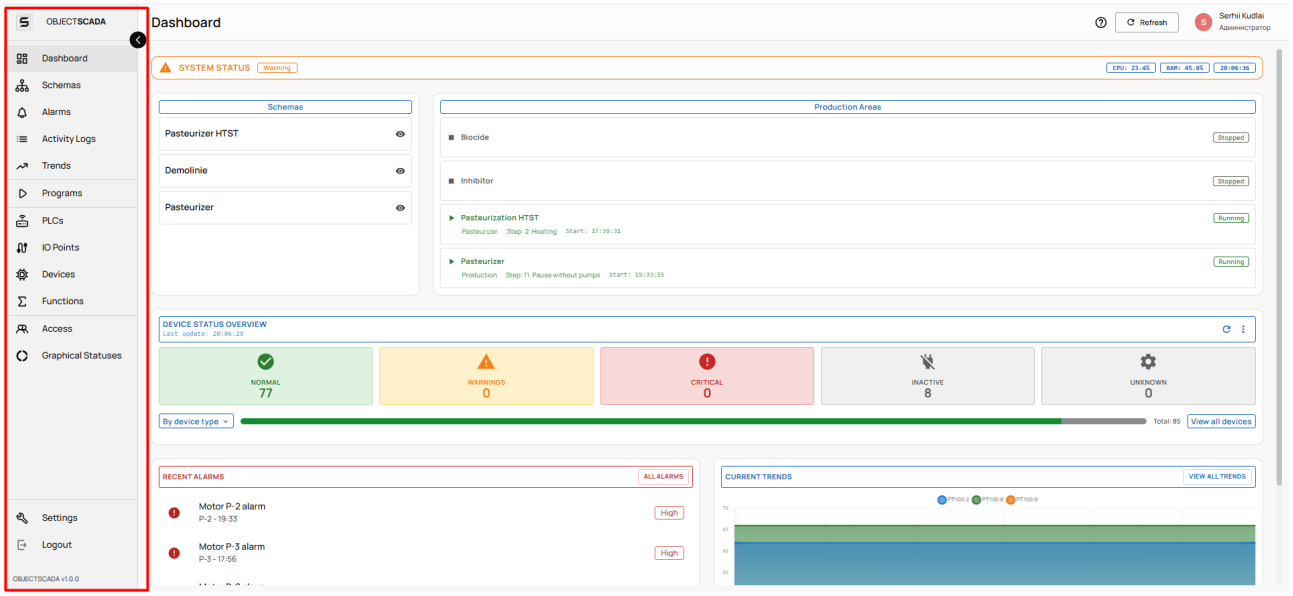
For users

The sidebar is available on every page. Menu items show or hide depending on the user's permissions.

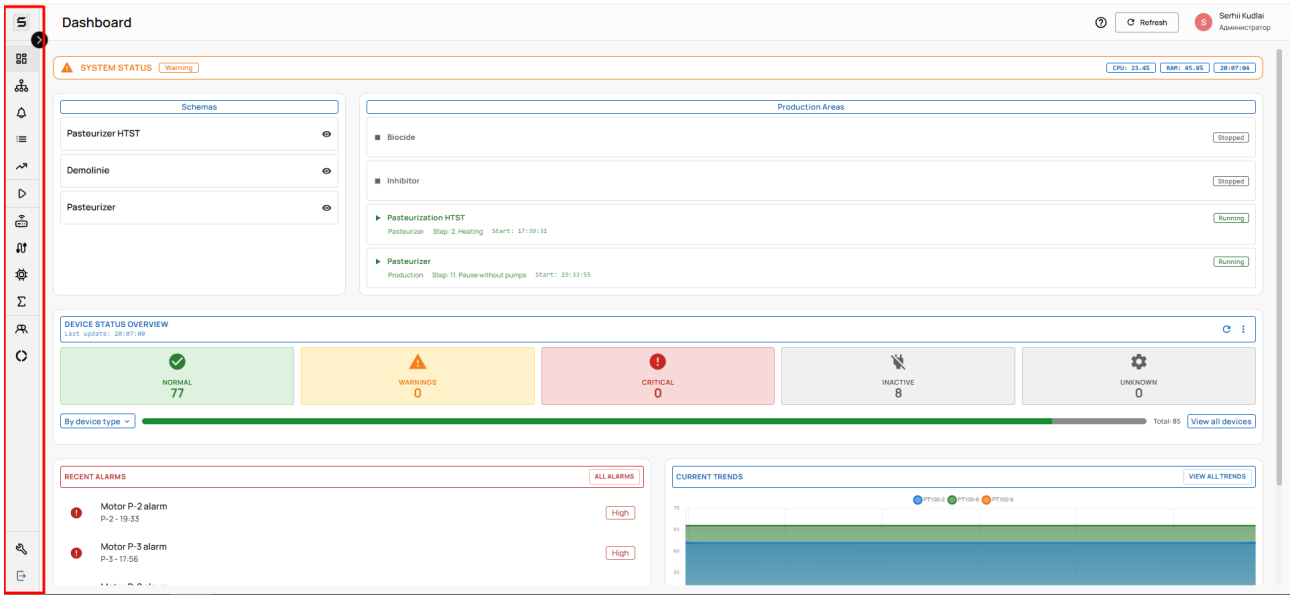
Menu item	Page	Required permission
Dashboard	Main panel	— (always available)
Schemas	Mnemonic schemas	—
Alarms	Alarms	—
Activity Logs	Activity log	—
Trends	Trends	—
Sequences	Sequences	sequences_recipes
PLCs	PLCs	plcs
I/O Points	I/O points	io_points
Devices	Devices	devices
Functions	Math functions	devices
Users	Users	users
Roles	Roles	roles
Graphical Statuses	Graphic statuses	graphical_statuses
Settings	Settings	— (always available)
Logout	Sign out	—

Additional sidebar elements:

Element	Description
User avatar	Photo and name
Collapse/Expand	Shrink the sidebar to icons
Version number	Current application version



Sidebar expanded: user avatar on top, menu items with icons and labels, Logout at the bottom.



Sidebar collapsed: only the menu icons.

Global UI elements

For users

Header:

Element	Description
Page title	Current section name
Help button	Launches the guided tour
Updates indicator	Badge when updates are available
Connection indicator	WebSocket/MQTT connection status
Contextual actions	Buttons specific to the current page

Snackbars:

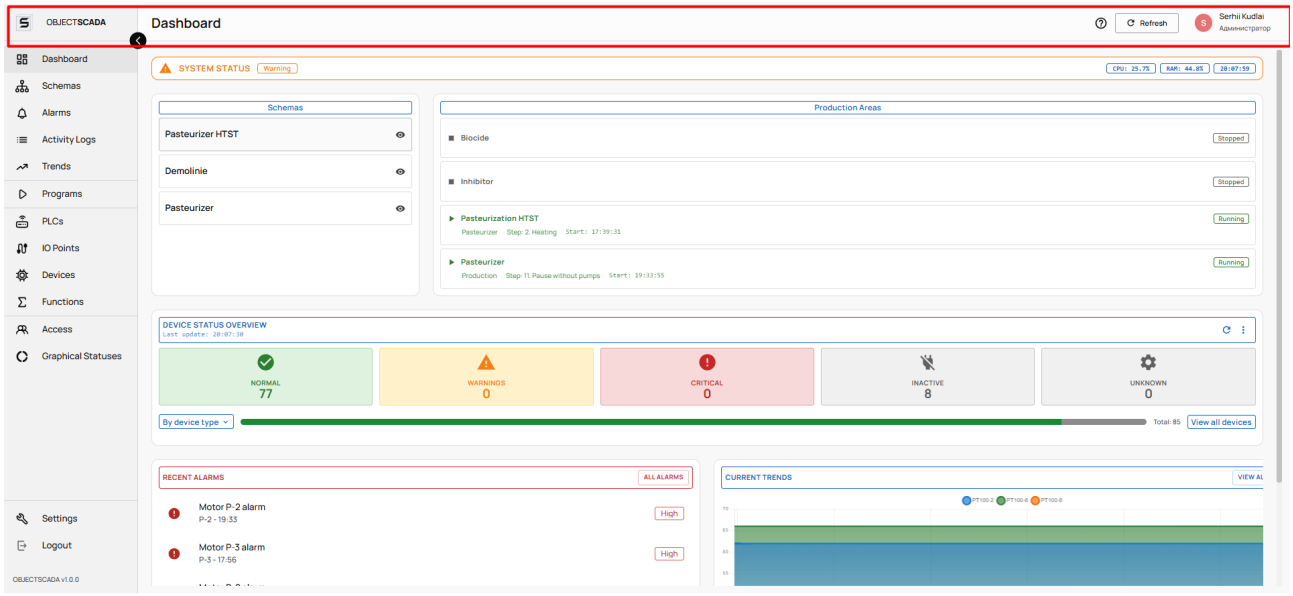
Type	Description
Success (green)	Operation succeeded
Error (red)	Error
Warning (yellow)	Warning
Info (blue)	Info

Banners:

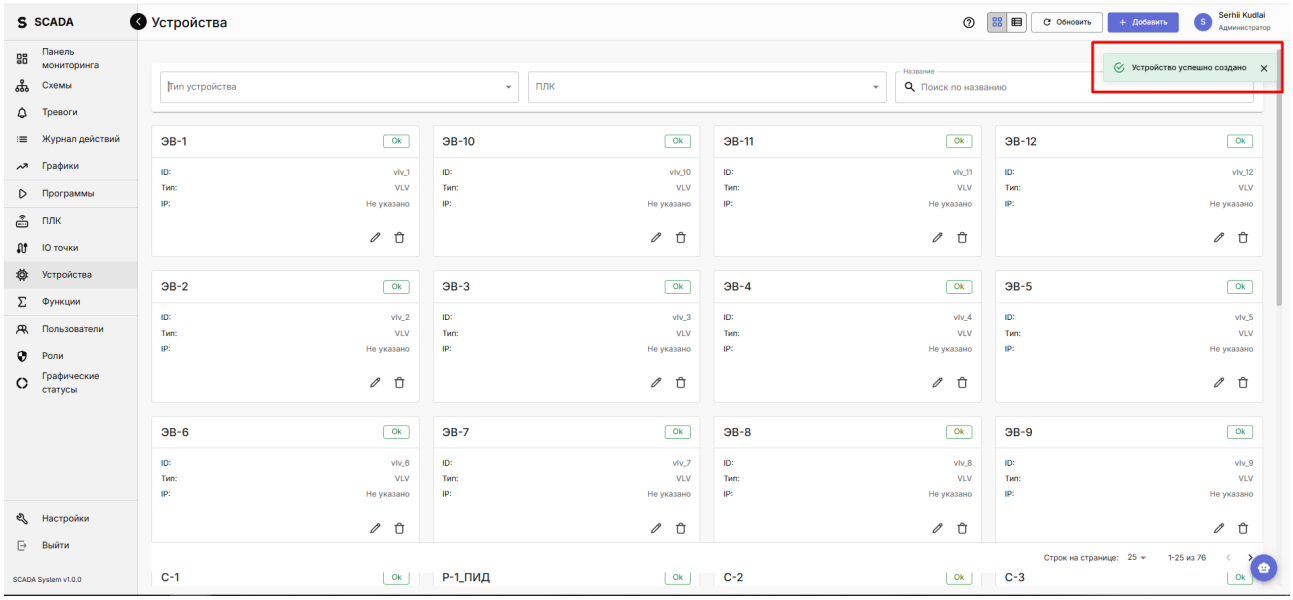
Banner	Description
Connection Status	Yellow banner when the connection is lost (after 15 s)
Maintenance	Blue banner when maintenance mode is on
Version Mismatch	Orange banner when frontend/backend versions differ
Update Available	Green badge in the header when an update is available

Guided tours:

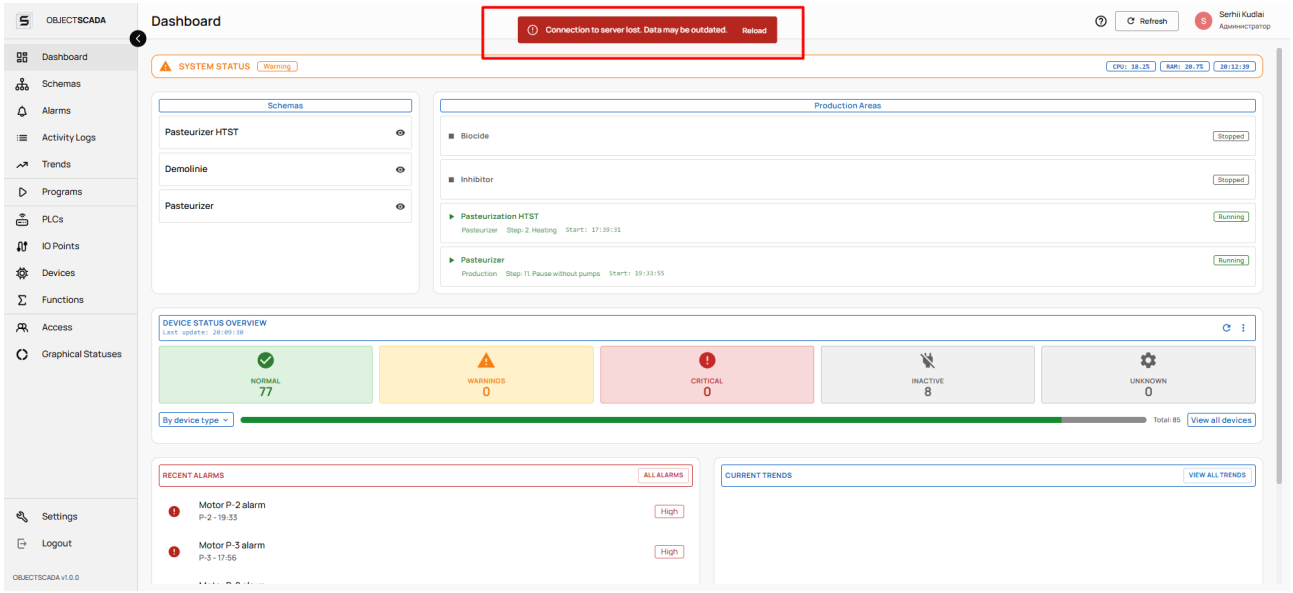
Available on every page via the «?» button in the header. Step-by-step interactive guidance with element highlighting and descriptions.



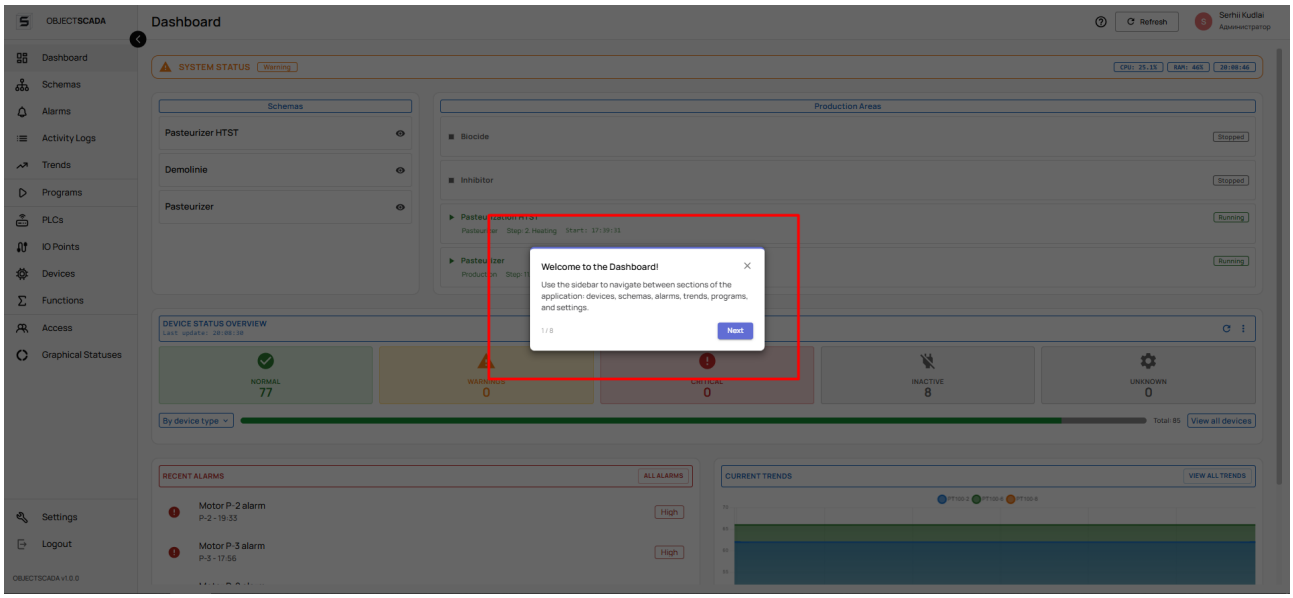
Application header: page title on the left, tour button (), updates indicator (red badge), connection status (green/red dot).



Sample notifications: green «Device saved successfully», red «Error: Connection lost», yellow «Warning: License expires in 7 days».



Yellow top banner: «Connection lost. Attempting to reconnect...» with animation.



Guided tour: highlighted element with a tooltip, «Next», «Back», «Skip» buttons, progress bar.

Keyboard shortcuts

Key	Action	Where it works
Ctrl+Z	Undo	Schema Builder
Ctrl+Y	Redo	Schema Builder
Ctrl+S	Save	Schema Builder
Delete	Delete element	Schema Builder
Ctrl+A	Select all	Schema Builder
Ctrl+C	Copy	Schema Builder
Ctrl+V	Paste	Schema Builder
Escape	Clear selection / Close dialog	Everywhere
Mouse wheel	Zoom	Schema Builder / Viewer, Trends

Practical examples

This section contains end-to-end automation scenarios that show how devices, mnemonic schemas, sequences, recipes and events come together for real processes.

Example 1. Automating a milk pasteurizer

Process goal. Heat milk to 75 °C, hold it for 15 seconds and cool it down to 4 °C. This is the classic HTST (High Temperature Short Time) process, widely used in the dairy industry.

Process equipment:

Device	ID	Type	Purpose
Supply pump	mot_00	Motor	Pushes milk through the heat exchanger
Steam inlet valve	val_001	Valve	Supplies steam to the heating section
Chiller valve	val_002	Valve	Supplies chilled water
Heating temperature sensor	ai_001	Analog Input	Temperature after the heating section
Holding temperature sensor	ai_002	Analog Input	Temperature after the holding tube
Outlet temperature sensor	ai_003	Analog Input	Temperature after the chiller
Heating controller	pid_001	PID Controller	Keeps heating temperature (setpoint → val_001)
Cooling controller	pid_002	PID Controller	Keeps outlet temperature (setpoint → val_002)
Receiving tank level	tnk_001	Tank	Tank fill monitoring
3-way divert valve	val_003	Valve	Returns undercooked milk for reheating

Stage 1. Device creation

Create all devices via Devices → «+». For PID controller pid_001:

```
POST /api/pid-controllers/
{
  "id": "pid_001",
  "name": "Heating controller",
  "plc": 1,
  "setpoint": 75.0,
  "kp": 2.5,
  "ki": 0.1,
  "kd": 0.05,
  "output_min": 0,
  "output_max": 100
}
```

Tank with automatic volume calculation via a strapping table:

```
POST /api/tanks/
{
  "id": "tnk_001",
  "name": "Receiving tank",
  "plc": 1,
  "volume_method": "STRAPPING",
  "level_sensor": "ai_004",
  "alarm_hh": 95.0,
  "alarm_h": 85.0,
```

```
  "alarm_l": 15.0,
  "alarm_ll": 5.0
}
```

Stage 2. Sequence

A 5-step sequence describes the full pasteurization cycle:

#	Step	Purpose	Transition condition to next step
1	Idle	Waits for operator start	Option «Start»
2	Heating	Heat to setpoint	ai_001 >= 75.0 for 5 seconds
3	Holding	Hold at temperature	Timer 15 s
4	Cooling	Cool down to 4 °C	ai_003 <= 4.0
5	Complete	Finish	Return to Idle

Create the sequence via the API:

```
POST /api/sequences/
{
  "name": "Pasteurization HTST",
  "name_translations": {"en": "Pasteurization HTST", "ru": "Пастеризация HTST"},
  "description": "Heat 75°C → hold 15 s → cool 4°C"
}
```

Add the steps:

```
POST /api/sequence-steps/
{
  "sequence": 1,
  "order": 2,
  "name": "Heating",
  "name_translations": {"en": "Heating"},
  "next_steps": {"3": [10]}
}
```

In next_steps the key "3" is the target step order and the value [10] is the list of transition condition IDs.

Transition condition «temperature ≥ 75°C»:

```
POST /api/transition-conditions/
{
  "id": 10,
  "type": "device",
  "device_id": "ai_001",
  "content_type": "analoginput",
  "operator": ">=",
  "value": 75.0,
  "duration_seconds": 5
}
```

> Important. duration_seconds guarantees that the condition holds continuously for the specified time — this prevents false transitions caused by transient signal spikes.

Stage 3. Production recipe

We lift the product-dependent parameters into recipe variables so the same sequence can run different dairy products (whole milk, cream, skim).

Recipe variables:

```
POST /api/production-recipe-variables/
{
  "name": "target_heating_temp",
  "name_translations": {"en": "Target heating temperature"},
  "sequence": 1,
  "default_value": 75.0,
  "min_value": 70.0,
  "max_value": 90.0
}
```

```
POST /api/production-recipe-variables/
{
  "name": "holding_time_sec",
  "name_translations": {"en": "Holding time, seconds"},
  "sequence": 1,
  "default_value": 15.0,
  "min_value": 10.0,
  "max_value": 30.0
}
```

Recipe for whole milk:

```
POST /api/production-recipes/
{
  "name": "Whole Milk 3.5%",
  "sequence": 1,
  "variables": {
    "target_heating_temp": 75.0,
    "holding_time_sec": 15.0
  }
}
```

Recipe for cream (harsher regime):

```
POST /api/production-recipes/
{
  "name": "Cream 30%",
  "sequence": 1,
  "variables": {
    "target_heating_temp": 85.0,
    "holding_time_sec": 20.0
  }
}
```

When a recipe is activated, the pid_001.setpoint is bound to target_heating_temp automatically and the Holding timer is bound to holding_time_sec.

Stage 4. Accident events

Define two critical accidents:

1. Overheat (above 90 °C):

```
POST /api/accident-events/
{
  "name": "Overheat protection",
  "device_id": "ai_001",
  "content_type": "analoginput",
  "operator": ">",
  "value": 90.0,
  "severity_level": 4,
  "action": "STOP_SEQUENCE"
}
```

severity_level: 4 is CRITICAL — the sequence is halted immediately.

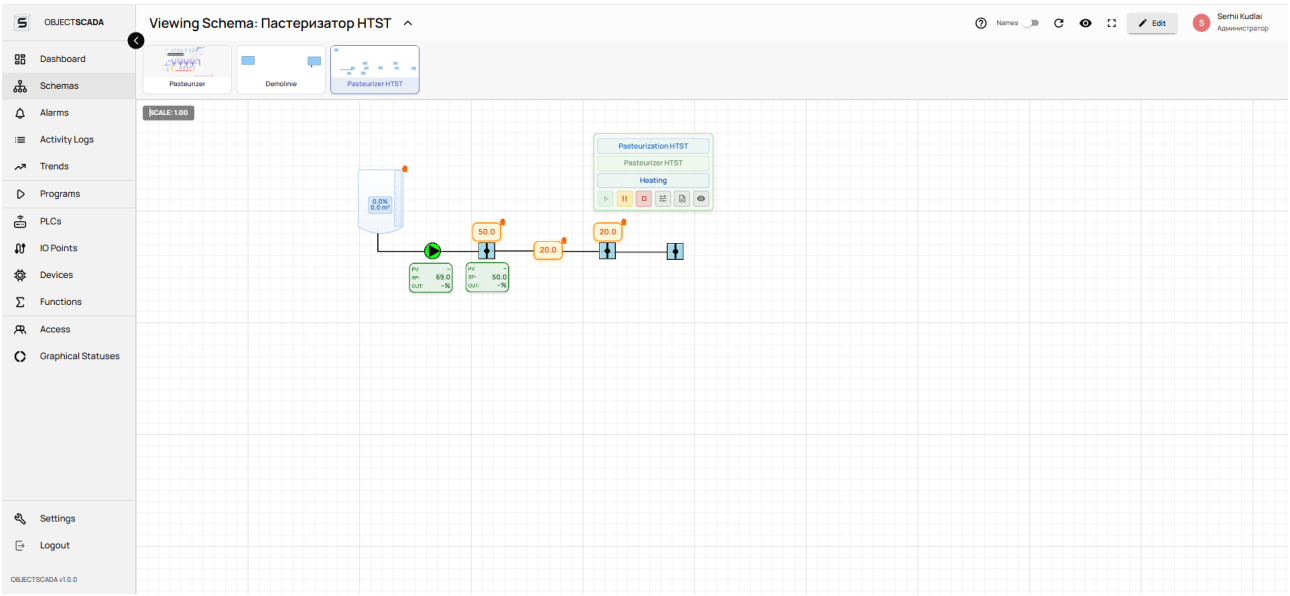
2. Under-pasteurization (below 73 °C during Holding):

```
POST /api/accident-events/
{
  "name": "Insufficient pasteurization",
  "device_id": "ai_002",
  "content_type": "analoginput",
  "operator": "<",
  "value": 73.0,
  "severity_level": 3,
  "action": "TRIGGER_DIVERT"
}
```

When this accident fires, the system activates divert valve val_003 automatically — under-treated milk is routed back for reheating.

Stage 5. Mnemonic schema

- Create a schema «Pasteurizer HTST».
- In Schema Builder place:
- Receiving tank tnk_001 via Tank — pick shape FLAT / DOME and dimensions 80×120 in the constructor.
- Pump mot_001, valves val_001, val_002, val_003, PIDs pid_001, pid_002.
- Sensors ai_001, ai_002, ai_003.
- Connect with pipes: tank → pump → heater (with val_001 and ai_001) → holding tube (ai_002) → chiller (val_002, ai_003) → outlet.
- Bind the sequence: Sequence → pick «Pasteurization HTST».
- Add option buttons «Start», «Pause», «Stop» (the separate Option block).



Pasteurizer mnemonic schema: receiving tank on the left with its level, a pipe runs to the pump, then through the heating section with a temperature sensor and a steam valve, the holding tube, and the chiller section with a chilled-water valve. On the right — the sequence block with the active «Heating» step highlighted.

Stage 6. Running the process

- Open the schema in Schema Viewer.
- On the sequence block pick the active recipe — e.g. «Whole Milk 3.5%».
- Press Start — the sequence moves from Idle to Heating.
- Watch the process live:
- Active step is highlighted (yellow → green on a successful transition).
- Current temperature on the sensor is visible in its popup.
- Percent open of val_001 (driven by PID).
- Once 75 °C is reached and held for 5 s, the transition to Holding happens automatically.
- After 15 s — transition to Cooling.
- When outlet temperature ≤ 4 °C → Complete → back to Idle.

Stage 7. Post-batch analysis

- Open Trends → create a chart.
- Add trend lines: ai_001, ai_002, ai_003, pid_001.output.
- Set the range to the last 30 minutes.
- Check:
- Shape of the heating curve (time-to-setpoint).
- Stability of the holding temperature (expect ≤ 0.5 °C ripple).
- Cooling rate.
- Export the chart to PNG or CSV for reporting.

Stage 8. Generating the PLC project

- Open Dashboard → PLC Export button (or Settings → PLC Export).
- In the dialog pick:
- PLC: PLC_DEMO
- Devices, sequences, recipes and accidents — all.
- Press Preview — the system lists the files to be generated:
- Motor_FB.scl, Valve_FB.scl, PID_FB.scl — device function blocks

- Sequence_Pasteurization_FB.scl — state machine
- Accident_FB.scl — accident handlers
- OB1_Main.scl — main program
- Tags_Inputs.xml, Tags_Outputs.xml — tag tables for TIA Portal
- Validate — verifies configuration correctness.
- Generate — produces a ZIP for TIA Portal import.

Example 2. CIP cleaning with batch management

Process goal. Automatic cleaning-in-place of processing equipment (tanks, pipelines) after a production shift. Multi-stage: pre-rinse → alkali wash → intermediate rinse → acid wash → final rinse.

Notable features of this example:

- Uses Batch Management — every cleaning cycle is recorded as a batch.
- Uses a Production Line with stages.
- The operator can skip individual stages via options (kurzes Reinigen).
- Demonstrates timers and counters.

Stage 1. Equipment

ID	Type	Purpose
tnk_cip_01	Tank	Alkali solution tank (NaOH 2%)
tnk_cip_02	Tank	Acid solution tank (HNO 1.5%)
tnk_water	Tank	Recirculated water tank
mot_cip_pump	Motor	CIP circulation pump
val_cip_supply	Valve	Supply valve into the wash loop
val_cip_return	Valve	Return valve
val_alkali	Valve	Alkali supply valve
val_acid	Valve	Acid supply valve
val_water	Valve	Water supply valve
ai_cip_temp	Analog Input	Solution temperature
ai_cip_cond	Analog Input	Conductivity (concentration check)
cnt_cip_flow	Counter	Supply flow meter
tim_cip_stage	Timer	Current-stage timer

Stage 2. Production line and stages

Create a Production Line with five stages — each stage will have its own sequence.

```
POST /api/production-lines/
{
  "name": "CIP Line 1",
  "description": "Cleaning of production 1 tanks"
}
```

Stages (ProductionLineStage):

#	Name	Sequence	Batch transfer action
1	Pre-rinse	seq_cip_rinse (water, 90 s)	TRANSFER to the next stage
2	Alkali wash	seq_cip_alkali (NaOH, 20 min at 75 °C)	TRANSFER
3	Intermediate rinse	seq_cip_rinse	TRANSFER
4	Acid wash	seq_cip_acid (HNO, 15 min at 65 °C)	TRANSFER
5	Final rinse	seq_cip_rinse	COMPLETE

The batch_transfer_action field on the last step of each sequence controls the transition:

- TRANSFER — the batch is moved to the next stage automatically.
- COMPLETE — the batch is completed.
- NONE — the batch stays on the current stage (manual control).

Stage 3. «Alkali wash» sequence

Steps:

- Fill — fill the loop with alkali (val_alkali = OPEN, mot_cip_pump = ON).
- Heat — heat to 75 °C (ai_cip_temp >= 75.0).
- Circulate — circulate for 20 min (timer tim_cip_stage).
- Drain — drain (val_cip_return = OPEN to sewer).

Recipe «Alkali wash — standard»:

```
POST /api/production-recipes/
{
  "name": "Alkali wash — standard",
  "production_line": 1,
  "variables": {
    "alkali_temp": 75.0,
    "alkali_duration_min": 20,
    "acid_temp": 65.0,
    "acid_duration_min": 15
  }
}
```

Note: the recipe is bound to the production line (production_line: 1), not to a sequence. The variables are available across all sequences in the line's stages.

Stage 4. Operator options

Add option buttons for speeding up or adjusting the process:

Option	Description	Effect
SKIP_ACID	Skip the acid wash	Stage 4 is skipped (jump straight to stage 5)
DOUBLE_RINS	Double final rinse	Stage 5 is repeated twice
LOW_TEMP	Mild mode (lower temperature)	Variables alkali_temp = 60, acid_temp = 50

Options are added through Sequence Options and placed on the schema as buttons.

Stage 5. Creating and starting a batch

- Open the CIP line schema in Schema Viewer.
- On the Production Line element press Create batch — the CreateBatchDialog opens.
- Fill in:

- Batch name: CIP-2026-04-18-001
- Recipe: «Alkali wash — standard»
- Stage: Pre-rinse (initial stage)
- Priority: Normal
- Create — the batch joins the queue.
- Activate the batch from the BatchManagementPanel → Start.

API equivalent:

```
POST /api/production-lines/1/create_batch/
{
  "name": "CIP-2026-04-18-001",
  "recipe_id": 5,
  "starting_stage_id": 1,
  "priority": "NORMAL"
}
```

Stage 6. Monitoring the batch

The BatchManagementPanel shows:

- Current stage (e.g. «Alkali wash»).
- In-stage progress (countdown timer).
- Current readings of key sensors (temperature, conductivity, flow).
- Overall batch progress across all stages.

Stage transitions are stored in BatchStageHistory — the full timeline can later be retrieved:

```
GET /api/batches/{id}/history/
```

Stage 7. Successful wash validation

HACCP/ISO 22000 requires documentary proof of cleaning. Use a reverse accident event:

```
POST /api/events/
{
  "name": "CIP validation — conductivity check",
  "device_id": "ai_cip_cond",
  "operator": "<",
  "value": 0.5,
  "duration_seconds": 30,
  "action": "LOG_EVENT"
}
```

This event logs the moment when final-rinse conductivity drops below 0.5 mS/cm for 30 s — evidence that detergent has been fully flushed out.

The log is available in Activity Logs with sequence/batch filtering and can be exported to CSV for reporting.

New process setup checklist

Use this template when configuring any new process:

1. Design (before entering the system):

- [ ] P&ID diagram of the process is drawn
- [ ] List of devices with unique IDs is prepared
- [ ] All analog and digital signals are defined

- [ ] Operating logic is documented (steps and transition conditions)
- [ ] Alarm thresholds and actions are defined

2. Configure in SCADA:

- [ ] PLC created with a correct address
- [ ] All devices created and attached to the PLC
- [ ] PID parameters configured (Kp, Ki, Kd)
- [ ] Tank strapping tables configured (where applicable)
- [ ] Mnemonic schema created with devices and pipes placed
- [ ] Sequence created with steps
- [ ] Transition conditions created
- [ ] Recipe variables created
- [ ] At least one recipe with default values created
- [ ] Accident events configured with proper severity levels

3. Testing (before going to production):

- [ ] Every device can be controlled manually (Manual mode)
- [ ] Transition conditions fire correctly (use POST /api/transition-conditions/{id}/simulate/)
- [ ] Sequence passes all steps end-to-end in Manual
- [ ] Accidents fire and halt the process correctly
- [ ] Recipes apply — setpoints update
- [ ] Trends record data for every key variable
- [ ] PLC export passes validation without errors

4. Commissioning:

- [ ] Operators are trained
- [ ] Project backup has been taken (POST /api/project/export/)
- [ ] Roles and permissions are assigned (operator, shift manager, engineer)
- [ ] First batch is started on reduced parameters as a sanity check
- [ ] Workplace documentation is prepared

Troubleshooting and FAQ

This section covers typical issues and how to solve them.

Connection and authentication

Q: The yellow «Connection lost» banner keeps showing.

Possible causes:

- Rust worker or Django backend is down — run docker-compose ps; all 6 containers must be Up.
- Nginx does not proxy WebSocket — open DevTools → Network → WS. A close code of 1006 or 502 points to nginx. Check nginx.conf: proxy_http_version 1.1 plus Upgrade/Connection headers are required.
- JWT token expired — sign out and back in. Under normal conditions the token refreshes automatically 1 minute before expiry.
- A network firewall blocks port 8001 (Rust WebSocket). The system falls back to Django WebSocket but with added latency — open the port.

Q: After an upgrade an orange «Version mismatch» banner appears.

Frontend and backend are at different versions. Refresh the browser with cache bypass (Ctrl+Shift+R). If that does not help, check docker-compose.yml: the frontend image tag must match backend and scada-worker.

Q: I cannot log in — «Invalid credentials».

- Check keyboard layout (CAPS LOCK).
- Make sure you are using the username, not the email (if the setting requires a specific one).
- If the password is lost and SMTP is not configured — reset it on the server: `bash docker-compose exec backend python manage.py changepassword `
- Check that your IP is not blocked in IP Access (an administrator might have enabled a whitelist).

Schema and devices

Q: A device on the schema stays grey and its values do not update.

- PLC offline — check the status in PLCs. If it is not Online, the device gets no data.
- Device not attached to the PLC — open the device card, check the PLC field.
- WebSocket channel not subscribed — check DevTools → Network → WS: there should be a ws/schema/{id}/devices connection. If not, reload the page.
- You lack permission on the device — ask an administrator to review your roles.

Q: A tank shows volume 0 although the level sensor works.

- Strapping table is not configured (for STRAPPING method). Open the tank → «Strapping Table» tab → add at least 2 points (sensor_value → volume).
- Wrong calculation method — check volume_method: STRAPPING or GEOMETRIC. GEOMETRIC requires TankSection entries (cylinder, cone, etc.).
- Rust worker is not updating the volume — check docker-compose logs scada-worker | grep tank_volume. By default the volume is computed every second (TANK_VOLUME_INTERVAL_MS).

Q: A pipe does not animate flow even though the motor is running.

Turn on Process Monitoring for the pipe in the right-hand property panel of Schema Builder. It is off by default.

Q: When I drag a device it jumps back.

This happens when a WebSocket update arrives at the same time. If you are in Schema Builder, editing blocks live updates; save immediately after the move (Ctrl+S).

Sequences and recipes

Q: The sequence does not transition to the next step even though the condition is met.

- Condition not bound to the step — open the current step, check next_steps. The target step must list the condition ID.
- `duration_seconds` has not elapsed — the condition must stay true continuously for the configured time. Look at the «Continuous for» field in the condition popup.
- The condition is simulated with another value — check the is_simulated flag in the WebSocket condition payload. Reset with POST /api/transition-conditions/clear_simulation/ and {"step_id": }.
- Device value is not being delivered — device offline (see above).

Q: I changed the recipe but the PID setpoint did not change.

Make sure that:

- The recipe is activated — POST /api/production-recipes/{id}/select-production-recipe/.
- The recipe variable is bound to a device field in the sequence editor (Steps tab → check device values per step).
- The sequence is not in is_pause = true — pause freezes all transitions and recipe applications.

Q: I want to test the logic without real equipment.

- Use transition condition simulation — POST /api/transition-conditions/{id}/simulate/ with {"simulated_value": 80.0}. The condition evaluates the supplied value instead of the real one.
- For analog inputs, enable Simulation mode: POST /api/analog-inputs/{id}/set_simulation/ with {"is_simulated": true, "simulated_value": 75.0}.
- After testing, always turn simulation off — it is dangerous in production.

Batches

Q: A batch hangs on a stage and does not move on.

- Check batch_transfer_action on the last step of the current stage's sequence — it must be TRANSFER (to move on) or COMPLETE (to finish).
- Check that the next stage exists and is active (is_active: true).
- Check the Rust worker log — a batch transfer failed message explains the cause.
- Manual transfer: POST /api/batches/{id}/transfer/ with {"next_stage_id": }.

Q: The batch queue order is wrong.

Sorting is by priority first, then by creation time. Change priority: POST /api/batch-queue/{id}/set_priority/ with {"priority": "HIGH"}. Valid values: LOW, NORMAL, HIGH, URGENT.

PLC export

Q: Pressing «Validate» produces a list of errors.

Common validation errors:

- Device ID contains invalid characters — the ID must match [a-z0-9_] only, no dashes or spaces.
- Sequence references missing device — the device was deleted but a reference in a step remains. Open the sequence editor and fix.
- PID output_max <= output_min — check the PID bounds.
- Tank strapping entries must be monotonic — sensor_value must strictly increase.
- No PLC assigned to device X — attach the device to a PLC.

Q: The generated SCL does not compile in TIA Portal.

- Check the TIA Portal version — the generator targets TIA Portal v16+.
- Make sure both XML tag tables (Inputs and Outputs) are imported before the SCL.
- Read the TIA Portal compilation log — it points at the specific FB and line.

Trends

Q: The chart is empty even though the device is working.

- Time range too small or in the future — reset to «Last 1 hour».
- Timescale offline — check GET /api/graphs/timescale-status/. If it returns disconnected, TimescaleDB is down or unreachable.
- No data points collected — in the Data Points section the device must have an active collection subscription.

Q: Trends are slow on large ranges.

Downsampling is used. If there are still too many samples, shrink the range or increase the aggregation interval. For long-term analysis export to CSV and process externally.

AI assistant

Q: The AI assistant returns «Authentication failed».

- Check the OpenRouter API key in Settings → AI Assistant. Keys are encrypted with Fernet on save — the error means either an expired key or a corrupted DB entry.
- Check your OpenRouter balance — when funds run out, requests are rejected.
- Check the logs: docker-compose logs ai-assistant | tail -50.

Q: The AI is slow.

The model runs an agent loop for up to 15 iterations with tool calls. Complex requests («analyse all accidents from last week and suggest optimisations») may take 30–60 s. For fast answers keep questions short and specific.

Performance

Q: The UI lags on a schema with 100+ devices.

- Disable flow animation on pipes that do not participate in the active process.
- In Settings → Performance enable «Throttle non-critical updates» (non-active devices update every 2 s instead of 250 ms).
- Give Docker more RAM (4 GB minimum for comfortable operation).

Q: The database grows quickly.

Main sources are the activity journal and historical trends. Configure retention in Settings → Data Management:

- Activity Logs: 90 days by default.
- Trend data: configurable per device via Data Points.

Safety and best practices

This section collects recommendations for safe system operation. Following them reduces incident risk and helps comply with quality standards (ISO 9001, HACCP, GMP).

Working with Auto / Manual modes

Manual mode gives the operator direct control bypassing sequences and alarm interlocks. It is a powerful tool — and a dangerous one.

Correct uses of Manual:

- Only for commissioning and diagnostics.
- Only on stopped equipment or in states explicitly designed for it.
- With verbal approval from the shift manager on a running line.
- With an entry in the activity log (this happens automatically, but the operator should know).

Incorrect uses of Manual:

- Bypassing safety conditions to speed up the process.
- Starting equipment manually without knowing the current batch state.
- Leaving equipment in Manual at the end of a shift.

> Rule. When you finish working with a device in Manual, always put it back to Auto. The system does not do it for you.

Simulation: tests only

Simulation of conditions, analog inputs and timers is a tool for the engineer, not the operator.

Scenario	Allowed?
Testing sequence logic on a test bench	Yes
Verifying accident firing in a simulator	Yes
«Skipping» a condition during production to start faster	Absolutely not
Simulating a sensor because the real one broke	No — fix the sensor or stop the process

Any live simulation during production is a potential cause of an incident. Before launching a production batch run the check GET /api/transition-conditions/?is_simulated=true — the list must not contain anything related to the active sequence.

Backups and recovery

What to back up:

- Project configuration — devices, schemas, sequences, recipes, accidents. Use POST /api/project/export/ → ZIP.
- Database — full PostgreSQL dump (contains journal, historical trends, batch data): `bash docker-compose exec postgres pgdump -U scada scadadb > backup_$(date +%Y%m%d).sql `
- User uploads — the ai_uploads volume (images for AI analysis).
- nginx and docker-compose.yml — keep under Git.

How often:

- Project configuration — before any significant change.
- Database — daily automated + before system upgrades.
- Before upgrading the SCADA version — both backup types are mandatory.

Test your restores. Once a quarter, restore a backup to a test stand and verify it. A backup that has never been restored is an unreliable backup.

Access control

Principle of least privilege. Every user gets only the permissions needed for their job.

Typical role matrix:

Role	Permissions
Operator	View schemas + control devices in Auto + acknowledge accidents. No edit access.
Shift manager	Everything the operator can do + start/stop batches + change recipes from an allowed list.
Technologist	Everything above + create/edit recipes and variables.
Automation engineer	Everything above + edit schemas, sequences, devices.
Administrator	Full access, including user management and system settings.

Also:

- Per-schema access — individual schemas can be hidden from users who do not work on that area.
- IP filtering — limit access to the internal network (192.168.0.0/16 or VPN).
- Regular audit — every quarter review the active user list in Users and delete leavers.

System upgrades

The correct upgrade sequence:

- Plan a window — upgrades require 5–15 min of production downtime.
- Take a backup (configuration + DB).
- Read the CHANGELOG — there may be breaking API changes or migrations.
- Run a dry-run — the system validates migrations without applying them.
- Apply the upgrade — via Settings → System → Updates.
- Wait for smoke tests — the system runs baseline checks automatically.
- If something breaks, use Rollback (available within 24 hours after upgrade).

> Never upgrade during an active batch or while unresolved critical accidents exist.

Handling accidents

What TO DO when an accident fires:

- Stay calm. Read the accident description in the Alarm Bar.
- Go to the equipment that triggered it.
- Visually assess the state (leak, overheat, unusual noise).
- If it is safe, press Acknowledge in Alarms.
- Remove the cause.
- Press Resolve when fixed.

What NOT TO DO:

- Mass-acknowledge all alarms without review (resolve-all is for exceptional cases, by shift-manager decision only).
- Delete accident events from configuration while production is running.
- Lower the severity level to silence a noisy alarm — instead, set correct thresholds.

Activity log

The activity log is your safety net and source of truth. It automatically records:

- Sign-ins and sign-outs.
- Every device control command (Start, Stop, Open, Close).
- Changes to recipes and sequences.
- Accident firings.
- Batch starts/stops/transfers.

Recommendations:

- Keep the log for at least 1 year (a requirement of many quality standards).
- Export to CSV regularly for archiving.
- When investigating an incident, first look at the log for the ±1 hour window around the event.

Naming recommendations

A consistent naming convention pays off as a project grows.

Devices:

- Prefix by type: mot_ (motor), val_ (valve), ai_ (analog input), pid_, tnk_ (tank).
- Zero-padded numbering: mot_001, mot_002 (up to mot_099).
- Group by area: val_tank1_in, val_tank1_out.

Sequences:

- Process prefix: seq_pasteurize_htst, seq_cip_alkali.

Recipes:

- Product name with parameters: Milk_Whole_3.5%, Cream_30%_HotFill.

Glossary

Reference for terms used in the system and this manual.

General automation

SCADA (Supervisory Control And Data Acquisition) — a class of software systems for monitoring and control of industrial processes.

PLC (Programmable Logic Controller) — an industrial computer executing control logic in real time.

HMI (Human-Machine Interface). In this system, the HMI is the Schema Viewer.

P&ID (Piping and Instrumentation Diagram) — a process diagram showing pipes and instrumentation.

I/O — Input/Output, the channels between the PLC and the field equipment.

Tag — a named PLC signal. In SCADA it maps to the memory address where a value is read/written (e.g. DB10.DBD0 for Siemens).

Devices

AI (Analog Input) — accepts a continuous signal (e.g. 4–20 mA, 0–10 V) and converts it to an engineering unit.

AO (Analog Output) — emits a continuous control signal (VSD setpoint, valve position).

DI / DO (Digital Input / Output) — a logical 0/1 signal (button, limit switch, lamp, contactor).

PID (Proportional-Integral-Derivative) — a control algorithm with three components: proportional (Kp), integral (Ki), derivative (Kd).

Setpoint (SP) — the target value of the controlled variable.

Process Value (PV) — the currently measured value of the controlled variable.

VSD (Variable Speed Drive) — a frequency converter controlling motor speed.

COS (Coordination Signal) — a logical variable used for cross-process synchronization (e.g. «line 1 ready for batch transfer»).

Counter — a pulse/event counter. Used for flow totalization, turns, cycles.

Interlock — a blocking condition (e.g. «do not start the pump while the valve is closed»).

Tanks

Strapping table — a table mapping level-sensor readings to tank volume. Used for tanks with non-standard geometry.

HH / H / L / LL — High High / High / Low / Low Low. Four alarm levels for tanks: critically high, high, low, critically low.

Hydrostatic level — a level derived from the pressure difference (ΔP) between upper and lower pressure transducers.

Sequences and recipes

Sequence — a state machine describing the ordered steps of a process.

Step — one state of a sequence.

Transition — a link between two steps with a trigger condition.

Transition Condition — the trigger: Device (device-value comparison), Timer (elapsed time), Option (operator button).

Recipe — specific device values for each step of a sequence.

Production Recipe — a set of variables for a specific product.

Recipe Variable — a parameter whose value depends on the product.

Option — an operator-driven control signal (Start / Pause / Skip button).

Batches and production lines

Batch — a production unit going through the process. The concept comes from ISA-88.

Production Line — a set of equipment that produces goods.

Production Line Stage — one processing stage on the line (e.g. wash → dry → pack).

Batch Transfer — the move of a batch from one stage to the next. Driven by the batch_transfer_action field on a sequence step.

Batch Queue — the queue of batches waiting to start on the line.

ISA-88 — the international standard for batch control in recipe-driven production.

PLC export and Siemens

TIA Portal — Totally Integrated Automation Portal, Siemens' PLC configuration IDE.

SCL (Structured Control Language) — the Siemens PLC language (IEC 61131-3 structured text).

FB (Function Block) — a program unit with logic and its own instance data.

DB (Data Block) — a memory structure for PLC variables.

OB (Organization Block) — an entry point for PLC code.

OB1 — the main cyclic OB, called every scan.

SimaticML — the XML format used for TIA Portal project export/import.

GSD file (Generic Station Description) — a hardware module descriptor for PROFIBUS/PROFINET.

WebSocket and API

REST API — Representational State Transfer over HTTP for CRUD operations.

WebSocket — a bidirectional TCP protocol enabling server-to-client push.

JWT (JSON Web Token) — an authentication token carrying user data in signed form.

SSE (Server-Sent Events) — one-way server-to-client streaming. Used by the AI assistant for streamed answers.

Delta update — sending only the changed fields of an object.

Batching — grouping several updates into one message to reduce network load.

Industry standards

HACCP (Hazard Analysis and Critical Control Points) — food-safety methodology.

GMP (Good Manufacturing Practice) — a manufacturing management standard, especially in pharma and food.

CIP (Cleaning In Place) — automatic cleaning of processing equipment without disassembly.

HTST (High Temperature Short Time) — high-temperature short-time pasteurization (72–75 °C, 15–20 s).

ISO 9001 — an international quality-management standard.

Appendix A: WebSocket channels reference

Channel	Description	Direction
ws/device/{id}	Device status	Server → Client
ws/device/{id}/control	Device commands	Bidirectional
ws/schema/{id}/devices	All devices of a schema	Server → Client
ws/schema/{id}/pipes	Schema pipes	Server → Client
ws/schema/{id}/sequences	Schema sequences	Server → Client
ws/schema/{id}/batches	Schema batches	Bidirectional
ws/schema/{id}/accidents	Schema accidents	Server → Client
ws/sequence/{id}	Sequence status	Server → Client
ws/sequences/all	All sequences	Server → Client
ws/pipe/{id}	Pipe status	Server → Client
ws/system	System metrics	Server → Client
ws/plc/status	PLC status	Server → Client
ws/device-status	Global status	Server → Client
ws/cos/all	Coordination signals	Server → Client
ws/math-functions	Math functions	Server → Client
ws/activity-log	Activity log	Server → Client
ws/accidents	All accidents	Server → Client

Rust WebSocket server (port 8001, proxied via `/rws/ws/`):

- Read-only for bulk device updates
- Automatic fallback to Django WebSocket
- JWT authentication in the URL query parameter

Appendix B: API reference (summary table)

Resource	Base URL	CRUD	Extra actions
Auth	/api/auth/	—	login, register, me, refresh
Users	/api/users/		set_password, upload-avatar, roles
Roles	/api/roles/		permissions
Permissions	/api/permissions/		categories, actions
Devices (all)	/api/devices/	—	alldevices, batchwebsocket_update
Analog Inputs	/api/analog-inputs/		setmode, setsimulation, setinterlock, setmanual_value
Analog Outputs	/api/analog-outputs/		(as above)
Digital Inputs	/api/digital-inputs/		(as above)
Digital Outputs	/api/digital-outputs/		(as above)
Motors	/api/motors/		(as above)
Valves	/api/valves/		(as above)
PID Controllers	/api/pid-controllers/		(as above)
Counters	/api/counters/		(as above)
Timers	/api/timers/		(as above)
COS	/api/coordination-signals/		(as above)
VSD	/api/variable-speed-drives/		(as above)
Tanks	/api/tanks/		calculatevolume, import/exportstrapping
Tank Sections	/api/tank-sections/		—
Tank Strapping	/api/tank-strapping-entries/		—
PLCs	/api/plcs/		status, set-run-mode
Schemas	/api/schemas/		add/remove devices, sequences, options, production lines
Pipes	/api/pipes/		—
Sequences	/api/sequences/		steps, transitions, positions
Sequence Steps	/api/sequence-steps/		—
Transition Conditions	/api/transition-conditions/		simulate, clear_simulation
Recipes	/api/recipes/		—
Production Recipes	/api/production-recipes/		select-production-recipe
Recipe Variables	/api/production-recipe-variables/		—
Events	/api/events/		set_active
Accident Events	/api/accident-events/		—
Accidents	/api/accidents/		acknowledge, resolve, resolve-all
Sequence Options	/api/sequence-options/		—
Production Lines	/api/production-lines/		activate, deactivate, create_batch
Production Stages	/api/production-line-stages/		—
Batches	/api/batches/		start, pause, resume, cancel, transfer
Batch Queue	/api/batch-queue/		reorder, set_priority
Pulse Modules	/api/pulse-modules/		start, stop

Resource	Base URL	CRUD	Extra actions
Math Functions	/api/math-functions/		calculate, validate, test
I/O Points (AI)	/api/points/analog-inputs/		validate, test
I/O Points (AO)	/api/points/analog-outputs/		(as above)
I/O Points (DI)	/api/points/digital-inputs/		(as above)
I/O Points (DO)	/api/points/digital-outputs/		(as above)
Graphical Statuses	/api/graphical-statuses/		upload-icon, list-icons
Graphs	/api/graphs/		data, timescale-status
Data Points	/api/data-points/		—
HW Catalog	/api/hw-catalog/		import-gsd
HW Stations	/api/hw-stations/		auto-assign-addresses
HW Slots	/api/hw-slots/		bulk-update
HW Network	/api/hw-network/		—
TIA Deploy	/api/tia/	—	deploy, auto-addresses, address-mapping
TIA Agent Config	/api/tia-agent-config/		check-connection, agent-status
Activity Logs	/api/user-activity-logs/	R/D	recent, statistics, clear
IP Access	/api/ip-access/allowed-ips/		checkip, myips
License	/api/license/	—	status, activate, generate
System	/api/system/	—	info, health, updates, maintenance
Project	/api/project/	—	export, import, backup, restore
PLC Export	/api/plc-export/	—	preview, validate, generate
AI Sessions	/ai-api/v1/sessions		messages
AI Chat	/ai-api/v1/chat	—	messages (SSE)
AI Images	/ai-api/v1/images	—	analyze, apply
AI Settings	/ai-api/v1/settings	R/U	—

Appendix C: Error and alarm codes

Reference of codes and statuses the user encounters — in logs, WebSocket payloads and API responses.

Accident severity levels

Field severity_level on AccidentType, or accident_event.severity_type.

Code	Name	Colour	Recommended response
1 / LOW	Low	Blue	Informational. No immediate action. Logged for statistics.
2 /	Mediu	Yellow	Requires operator attention soon. Production can continue.
3 / HIGH	High	Orange	Immediate attention required. Potential product-quality degradation or equipment-damage risk.
4 / CRITICAL	Critical	Red	Immediate sequence halt. Risk to people or equipment.

Event severity types (`SeverityType` for Events):

Code	Name	Use
INFO	Informational	Status messages (start, stage completion)
WARNING	Warning	Non-threatening deviation
ERROR	Error	Critical issue requiring intervention

Accident statuses

The status field on Accident. Lifecycle:

```
NEW → ACKNOWLEDGED → IN_PROGRESS → RESOLVED → CLOSED
```

Code	Name	Description
NEW	New	Just raised, unacknowledged. Shown in the Alarm Bar.
ACKNOWLEDG	Acknowledged	Operator confirmed receipt (POST /api/accidents/{id}/acknowledge/). Stays active but drops out of top priority.
IN_PROGRESS	In progress	Root-cause work in progress.
RESOLVED	Resolved	Root cause fixed (POST /api/accidents/{id}/resolve/).
CLOSED	Closed	Final state after RESOLVED + a delay (for reporting).

Sequence statuses

Fields on the Sequence model:

Field	Type	Meaning	Description
is_running	bool	true	Sequence is active, steps switch.
is_pause	bool	true	Paused; transitions are frozen, devices hold state.
is_stop	bool	true	Stopped; devices are brought to a safe state.

Combinations:

- is_running=true, is_pause=false, is_stop=false — running.
- is_running=true, is_pause=true, is_stop=false — paused (resumable).
- is_running=false, is_pause=false, is_stop=true — fully stopped.

Batch statuses

Code	Name	Description
PENDING	Pending	Queued, not started
RUNNING	Running	Active batch
PAUSED	Paused	Temporarily halted by operator
COMPLETED	Completed	Finished all stages successfully
CANCELLED	Cancelled	Aborted manually
FAILED	Failed	Aborted due to a critical accident

Priorities:

Code	Queue order
URGENT	Runs first
HIGH	2
NORMAL	3 (default)
LOW	4

Device modes

Applicable to motors, valves, PIDs, AI/AO/DI/DO:

Code	Description
AUTO	Controlled by sequence / system logic
MANUAL	Direct operator control
SIMULATION	Simulated value (testing only)
INTERLOCK	Locked by an interlock (value cannot be changed)

API error codes

Typical HTTP codes and their meaning in this system:

Code	Meaning	Typical cause
200 OK	Success	Request completed
201 Created	Created	New object created
204 No Content	Success, no body	Deletion
400 Bad Request	Invalid request	Validation failed. Body is {"field": ["error msg"]}
401 Unauthorized	Unauthenticate	JWT missing or invalid
403 Forbidden	No permission	Authenticated but not authorised

Code	Meaning	Typical cause
404 Not Found	Not found	ID does not exist
409 Conflict	Conflict	E.g. creating a device with an existing ID
423 Locked	Locked	Object locked by another process (e.g. saving a schema while a sequence is running)
500 Internal Error	Server error	Bug or exception. Check docker-compose logs backend
502 Bad Gateway	Proxy error	Nginx cannot reach backend / Rust worker
503 Service Unavailable	Unavailable	Maintenance mode or overload

WebSocket close codes

When a WebSocket connection closes, the following codes may be observed:

Code	Description
1000	Normal closure
1001	Server going away (restart)
1006	Abnormal closure (connection lost). Frontend auto-reconnects.
1008	Policy violation (e.g. invalid JWT in the URL query parameter)
1011	Internal server error
4001	Custom: Authentication failed
4003	Custom: Forbidden (no channel permission)
4004	Custom: Channel not found

Transition condition types

Code	Description
device	Based on a device value (compared to a threshold)
timer	Based on duration (after entering the step)
option	Based on an operator option press

Comparison operators (for type=`device`):

Symbol	Meaning
==	Equal
!=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal

Batch action on step (batch_transfer_action)

Code	Description
NONE	No action; batch stays on current stage

Code	Description
TRANSFER	Transfer batch to the next stage
COMPLETE	Finish batch (final status)

Tank cap shape types

For field SchemaDevice.configuration.tank_visual:

Bottom (`bottom_type`):

Code	Description
FLAT	Flat
DOME	Spherical (convex downward)
DISH	Torispherical (elliptical)
CONE	Conical

Top (`top_type`):

Code	Description
FLAT	Flat
DOME	Spherical
DISH	Torispherical

Document generated based on source-code analysis. For screenshot placeholders: open the application, navigate to the indicated screen and take a screenshot matching the described elements.