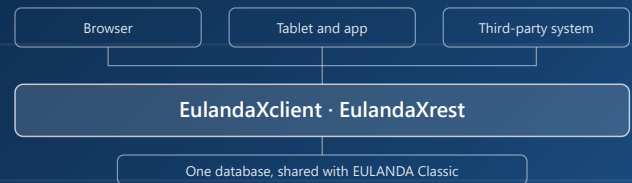


DATASHEET

EULANDA X-Suite



EULANDAXREST 1.0.5 · EULANDAXCLIENT 1.1.3 · AS OF 25 SEPTEMBER 2026

The new access layer for the EULANDA ERP system: a documented REST API, an MCP server for AI tools and a web interface with a built-in assistant for browser, tablet and desktop. On the very same database as EULANDA Classic.

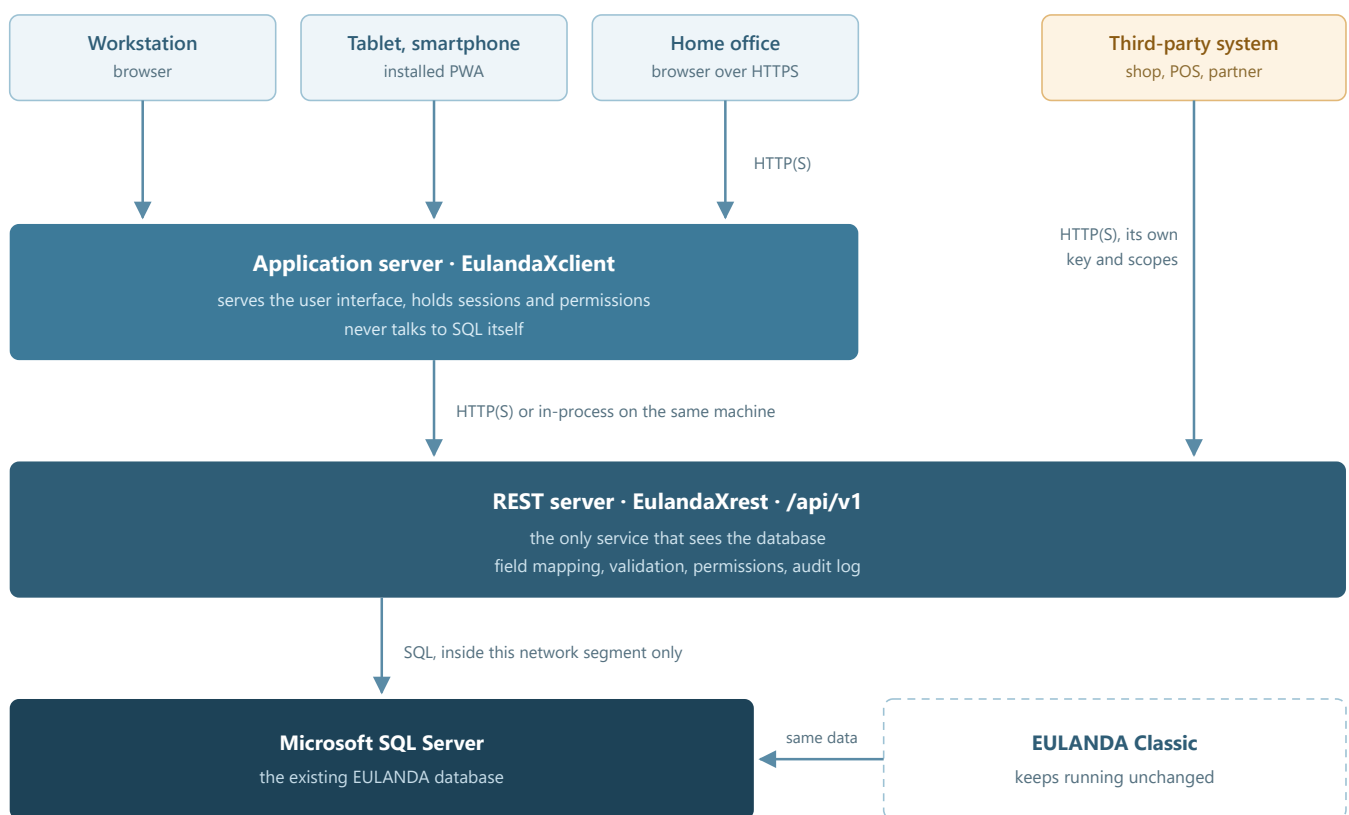
In short. EulandaX replaces the direct SQL connection of every workstation with exactly one service. Everything else speaks nothing but HTTP(S) - browsers in the office, a tablet in the warehouse, a branch office without VPN, an online shop from the outside. The data stays exactly the data of the existing EULANDA installation: no second database, no synchronisation.

- **Nothing has to be reinvented** - the commercial logic has always been calculated by the SQL Server. The API simply calls what has been in production for years.
- **No installation on the workstation** - a browser is enough, on tablets also as an installable app (PWA).
- **No VPN and no SQL access on the network** - only the REST server sees the database.
- **Runs alongside EULANDA Classic** - same database, same posting routines, replacement step by step.
- **Grows with you** - from a single workstation up to several servers behind a load balancer, with no change to the application.
- **Extensions without a fork** - your own modules are added as plugins, bringing their own menu entries, screens and permissions. The standard stays untouched and remains upgradeable.
- **REST, MCP and assistant** - next to the documented REST API the suite brings an MCP server through which AI tools work with the data, and an assistant right in the interface. It handles tasks such as "Create a quote for customer Mustermann with articles A, B and C."
- **In live operation** - since September 2026 in daily use at EULANDA itself and set up at the first customers. The interface is fully documented at sdk.eulanda.eu/rest.

For the decision, in five sentences. There is no cut-over day: both worlds work on the same data at the same time, and each department moves as soon as its screens are ready. There is nothing to roll out: whoever is to work in the browser gets an address, not a setup. No second calculation path appears, because the commercial logic stays where it has been for years - in the database. Your own adjustments remain your own modules and are not overwritten by updates. And operations get simpler rather than harder, because the database port no longer leaves the server room.

1. The principle

Traditionally every workstation needs its own SQL connection: driver, connection file, an open database port, and a VPN bridge for field staff. In EulandaX there is **exactly one service with database access**, the REST server. It publishes the data as endpoints (for example `/api/v1/addresses`). Every consumer fetches JSON there, changes it and sends it back.



Four layers, one set of data: only the REST server sees SQL, everything above it speaks HTTP(S).

The two server roles

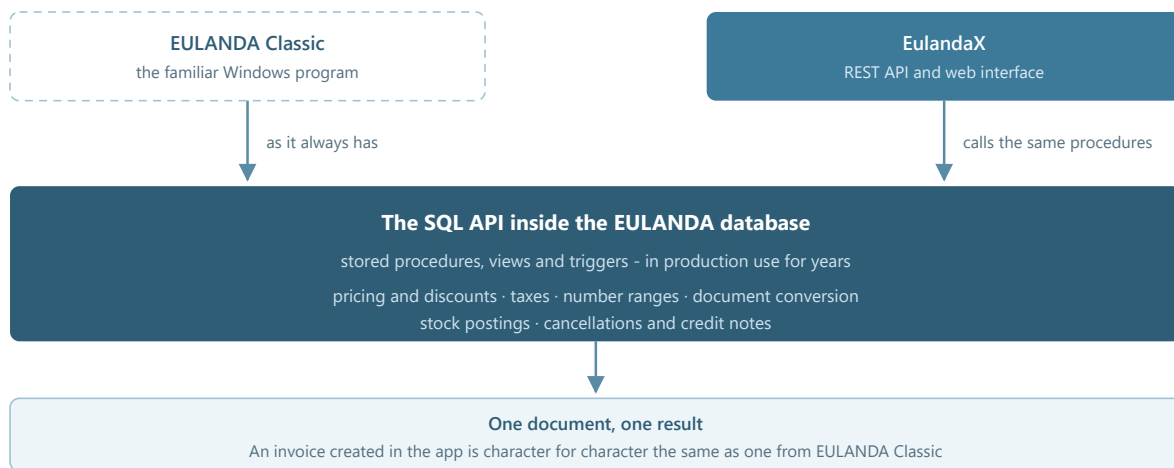
	EulandaXrest (REST server)	EulandaXclient (application server)
Purpose	data API <code>/api/v1/*</code>	serves and operates the web interface
Talks to	SQL Server	EulandaXrest and nothing else
State	stateless, every call carries its own credentials	holds device and login sessions
Location	on the SQL machine or in its LAN	anywhere with HTTPS access to the REST server
Default port	8100	8102

The browser **never** talks to the REST server directly; the application server makes that call on the server side. This keeps the API key on the server, where it never reaches a device. If both services run on one machine, the REST part runs **in the same process**: same code, same contract, just without the network hop. The very same user interface therefore works from a single workstation up to a multi-site installation without a single line of code being changed.

2. Nothing has to be reinvented

This is why the project does not take a decade: **the commercial logic of EULANDA has always lived in the database**, not in the Windows program. Prices, discounts and scales, taxes, number ranges, the conversion from quote through sales order and delivery note to invoice, stock postings, cancellations and credit notes - all of that is done by stored procedures and views inside SQL Server. EULANDA Classic is, at its core, a user interface operating that SQL interface.

The REST server therefore calculates nothing by itself. It validates the request, translates the field names and then calls exactly the same procedure the Windows program calls. There is no second calculation path that could drift apart over time.



Both front ends operate the same SQL interface. The API is a new way in, not a reimplementation.

What this means in practice:

- **No business rules to maintain twice.** If a rule changes in the ERP system, it applies in the new client immediately - nobody has to program it a second time.
- **No test effort for the calculations.** What gets tested is the way in, not the commercial logic. That logic runs in thousands of documents every day.
- **No deviations in the documents.** Number ranges, rounding and tax breakdown come from one single source, so two versions of the truth cannot arise.
- **Fast to extend.** A new resource in the API is essentially a mapping of English field names onto existing views and procedures, not the development of new business logic.

That is why the data layer is already finished. The REST API covers the scope of the ERP system today: addresses and contacts, articles with prices, stock, serial numbers and parts lists, categories, suppliers and purchasing, the document chain from quote through sales order and delivery note to invoice including cancellations and credit notes, warehouse, point of sale with fiscal signature, payments, document management, statistics, number ranges, company master data and the registry. All of it is documented and can be tried out directly: **sdk.eulanda.eu/rest**. What is being built now, step by step, are the **visual shells** on top of it - the screens of the web interface.

3. The user interface

One single HTML application serves every mode of operation. It is updated once on the application server and is therefore instantly current on every device - no rollout, no package distribution.

Desktop window

Starts like a Windows program (WebView2) and fetches its data through the same contract.

Browser on the network

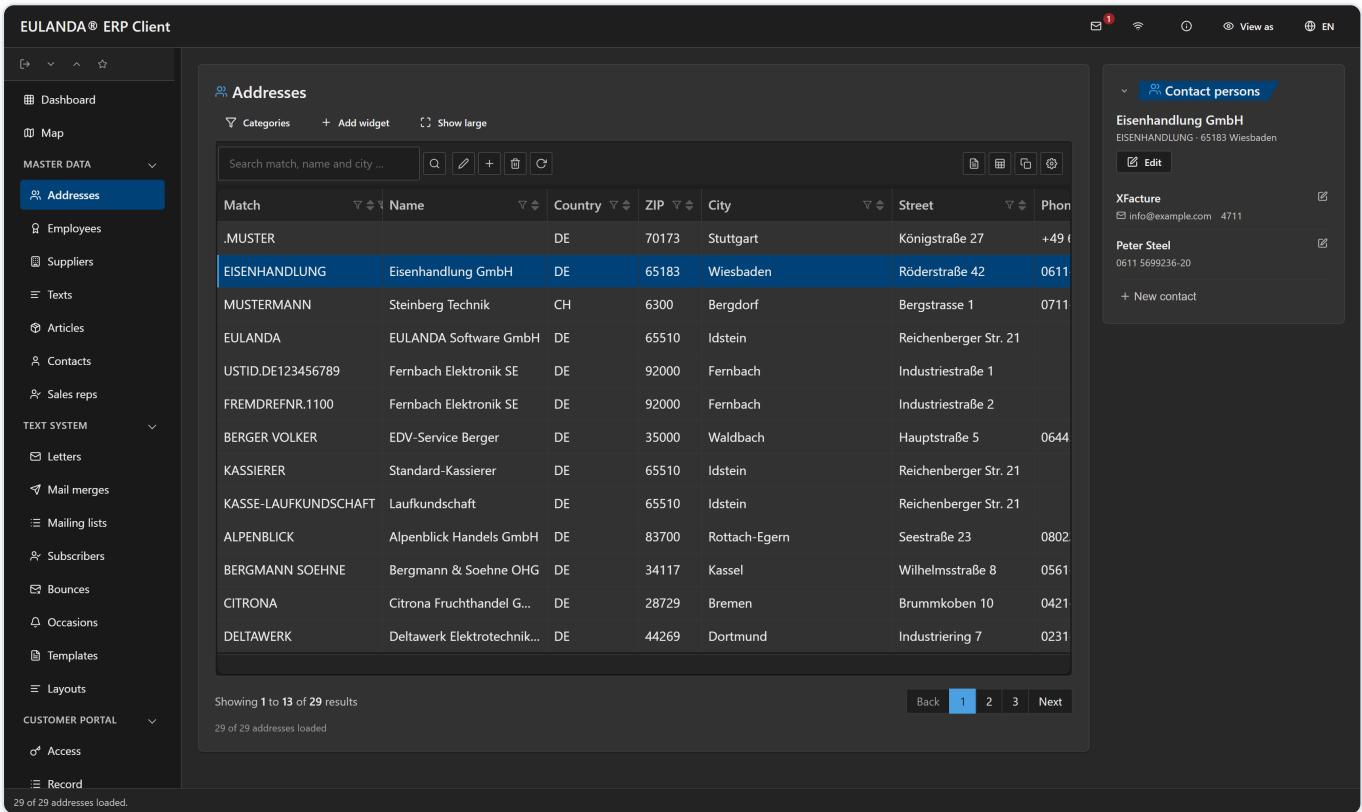
Open the address, done. Zero installation, from the home office as well.

PWA on a tablet

Icon on the home screen, its own window. Android, iPad, Windows tablet.

Third-party system

Shop, point of sale or partner software address the REST API directly with their own key.



Address list in the browser: search, category filter, contacts panel and the "view as" role switch

EULANDA® ERP Client

Dashboard Map MASTER DATA Addresses Employees Suppliers Texts Articles Contacts Sales reps TEXT SYSTEM Letters Mail merges Mailing lists Subscribers Bounces Occasions Templates Layouts CUSTOMER PORTAL Access Record

Address: EISENHANDLUNG

Record 2 of 29

Back Save Delete Settings Actions

General Extended

Identification

Match EISENHANDLUNG
Unique short code of the address (required for new records).

Ledger account 10200

Group

External reference 123456789-4711

External number 54321

Communication

Phone 0611-5699236-14

Mobile

Fax

Email einkauf@eisenhandlung.de

Website www.wiesbaden.de

Language German

Terms

Address

GLN 4317784008914

Name Eisenhandlung GmbH

Name 2 Hebebühnen aus ALU

Name 3 Industriefläche Süd

Street Röderstraße 42

City D 65183 Wiesbaden

Assignment

Sales rep 4

Commission (%)

Salutation

Overview

Eisenhandlung GmbH / Hebebühnen aus ALU / Industriefläche Süd

Match EISENHANDLUNG

Street Röderstraße 42

City DE 65183 Wiesbaden

Phone 0611-5699236-14

Email einkauf@eisenhandlung.de

VAT ID DE20215869

Categories

Daten Wartung

Gesperrt

Regionen

Shop

TEST

Image

29 of 29 addresses loaded.

The address screen with field help, category tree, notes and image - fields and visibility follow the user's role

Sign in

Please sign in with your user name and password.

User

Password

Sign in

Login with user name and password - shown here in the light theme

Light or dark is one click away and is remembered per device. By default the application follows whatever the operating system asks for. The remaining screenshots in this data sheet show the dark theme.

EULA...

Addresses

Categories Show large

Search match, name and city ...

Showing 1 to 10 of ... Back 1 2 3 Next

.MUSTER
10200
DE - 70173 - Stuttgart - Königstraße 27 - +49 6104 12345

EISENHANDLUNG
10200
Eisenhandlung GmbH - DE - 65183 - Wiesbaden - Röderstraße 42 - 0611-5699236-14

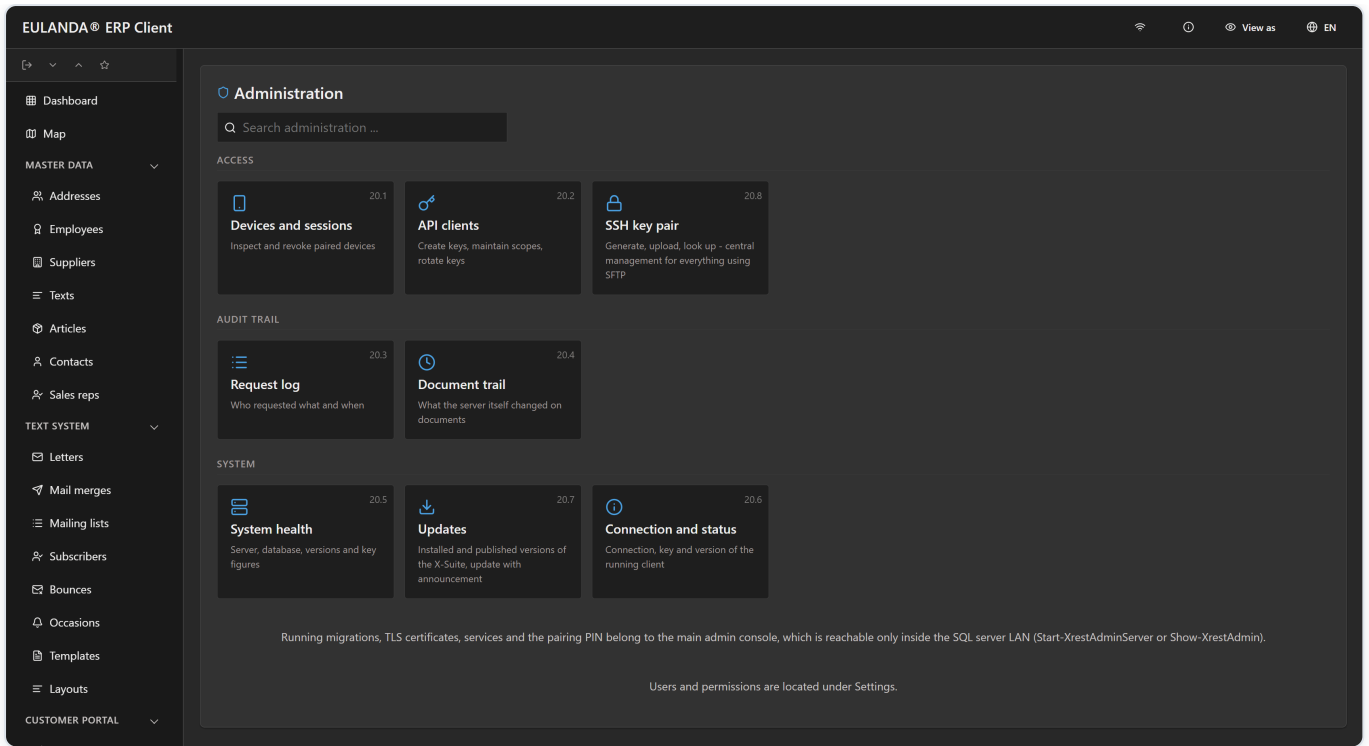
MUSTERMANN
10001
Steinberg Technik - CH - 6300 - Bergdorf - Bergstrasse 1 - 0711-000000

EULANDA
10035
EULANDA Software GmbH - DE - 65510 - Idstein - Reichenberger Str. 21

USTID.DE123456789
10034
Fernbach Elektronik SE - DE - 92000 - Fernbach - Industriestraße 1

29 of 29 addresses loaded.

The same application on a smartphone: the table turns into a card list automatically



Administration inside the client: paired devices, API keys with scopes, access log and system status

4. Scalability

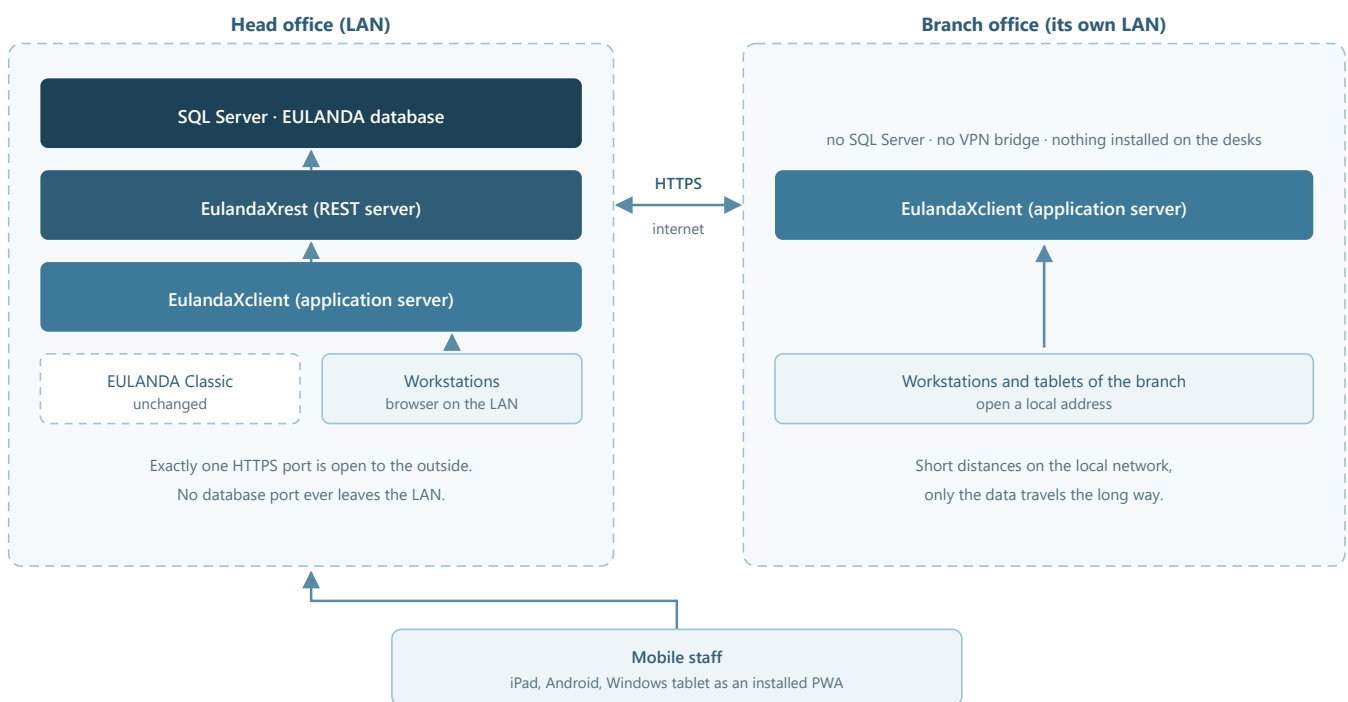
Concurrent processing. Both servers answer several simultaneous requests truly in parallel; a slow query does not block the others. Each request runs on its own database connection, checked for liveness - no connection setup per click. Concurrency is a deliberate switch, so a small installation stays lean and predictable. Measured on the test system: 24 requests one after another took 2818 ms, the same 24 in parallel 927 ms.

Growing horizontally. The REST server is stateless. If the load of many browsers and tablets outgrows one machine, you add further REST servers on the same database and distribute the requests through a balancer. The interface layer can be distributed as well, typically one application server per site.

Sessions survive restarts. Pairing and login are recorded in the tenant database (pairing 180 days, login 12 hours, both sliding). A server update therefore costs nobody a new PIN entry and no new sign-in.

Order of magnitude. One machine comfortably serves around 50 users per tenant. Beyond that you do not rebuild, you add - the application itself stays untouched.

Several sites, without VPN and without a second SQL Server



The branch needs neither a database nor a VPN. Its application server talks to the head office REST server over HTTPS.

If you prefer it even simpler, leave the local server out of the branch and open the head office address in the browser instead. Both ways deliver the same application; the local application server merely shortens the distances inside the branch network.

5. Security

- The API key is the trust boundary.** It is stored as a hash only (PBKDF2-SHA256) and shown exactly once, when it is created.

- **Scopes limit the routes.** A key holding `addresses:read` reads addresses and nothing else.
- **A role per key.** No access is an administrator by default; only explicitly trusted keys may act on behalf of a user.
- **Users, roles and permissions** live as data in the tenant database and are enforced on the server, not in the browser. Fields a role must not see never leave the server.
- **Device pairing by PIN** in server mode, with a lockout against brute force.
- **No SQL from the client.** Filters are named parameters against a field whitelist, and every value is passed as a parameter.
- **TLS through http.sys**, coexisting with an existing IIS on port 443. No IIS required, no additional web server.
- **Traceability:** every write is recorded in the access log with client, resource, key and result.

6. One set of data, no synchronisation

Just like the logic (section 2), the data is already there: EulandaX is not a new system with its own data storage, it is a new way into the existing one.

- **One database.** No second database, no synchronisation, no time lag, no conflicts - and therefore no interface that could run overnight and be missing in the morning.
- **Parallel operation without a switchover date.** Accounting keeps working in the familiar program while field service and warehouse already work in the browser - both see the same data at the same moment.
- **Existing add-ons keep working.** Everything built on the database today sees documents entered through the API exactly as it sees those from the Windows program.
- **Free of collisions.** What EulandaX needs for itself lives in its own database schema and is applied through numbered, repeatable migrations. The existing EULANDA update pipeline is untouched.

Replacement instead of migration. The new system can replace the old one department by department - with no migration project, no cut-over date and no risk, because both worlds work on the same data at any time.

7. Extending without a fork

Hardly any grown installation gets by without adjustments - extra fields, extra reports, an association requirement, a procedure that exists in this company only. Traditionally this ends in modified standard files, and at the next update either the adjustment is gone or the update is postponed. **EulandaX is built so that this does not have to happen.** Extensions are added as a **plugin** and sit entirely next to the standard - the core is not touched, nothing is overwritten.

A plugin gets its own namespace on every level:

Level	What belongs to the plugin
Database	its own schema with its own migrations. Core tables are never modified; core data is read, and writing goes through the existing core routes.
REST	its own path <code>/api/v1/<plugin>/*</code> with its own scopes. The plugin's interface description is merged into the server's documentation automatically.
Interface	a description in JSON from which the client builds menu entries, tabs, tables, forms and actions itself - using the same building blocks as the standard.
Permissions	its own permissions, merged into the role matrix. Menu entry and route hang on the same permission - invisible in the client also means rejected at the server.
Language	its own translations in its own key namespace, merged in when the page is built. Overwriting standard texts is ruled out.

Because everything lives in its own namespace, the life cycle is a short list: **install** (migrations, registration, permissions), **disable** (routes and menu disappear, the data stays) and **uninstall** (schema gone, registration gone, permissions gone) - without residue and per tenant. A core update cannot overwrite an extension, because it owns nothing of it.

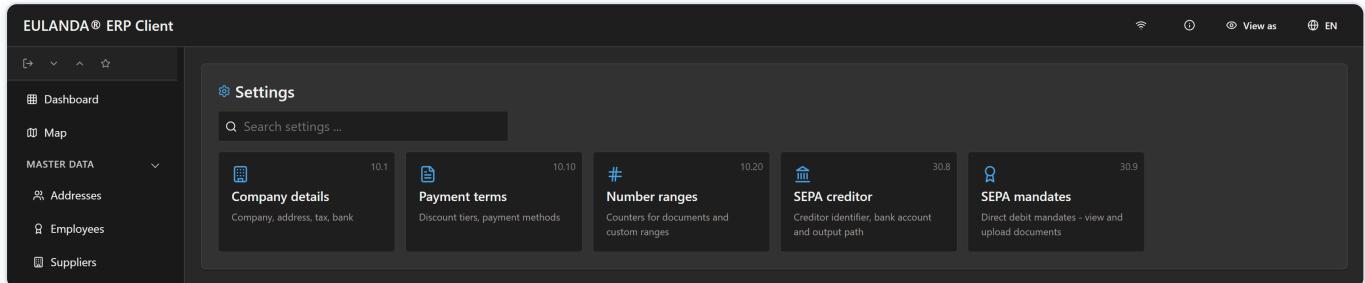
The plugin brings its own user interface

The interesting part is the interface: as a rule a plugin does not have to write **a single line of application JavaScript**. It describes what it wants to show, and the client renders it:

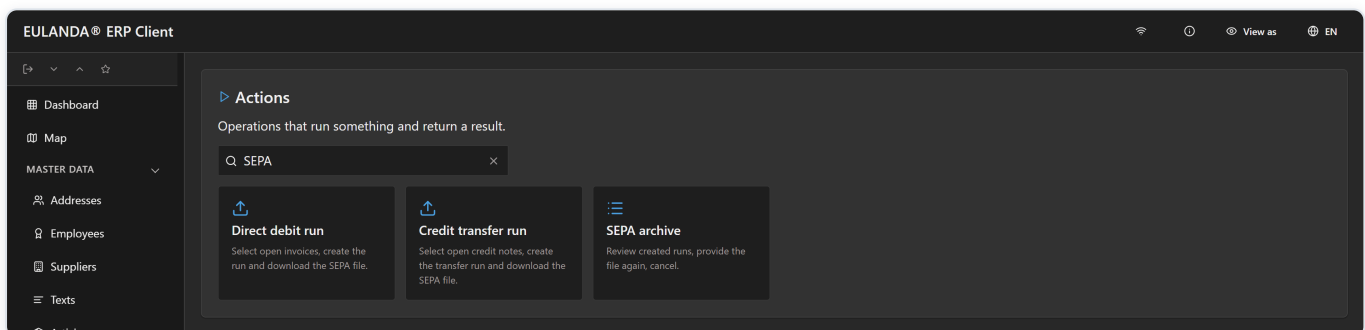
- **Its own menu entries** in the main navigation, with tabs beneath them.
- **Tables with a detail view** - click a row, look at the record, and the matching actions sit on the record (download again, cancel, and so on).
- **Its own settings tiles**, appearing next to the standard tiles in the settings area.
- **An actions area** for things that do something rather than display something: triggered runs, imports, exports - each with a result and a file.
- **Documents from the archive** right on the record: view, upload, remove - through the same routes the standard uses.

For genuine special cases (a map, a bespoke widget) there is a defined hook for a dedicated interface module. It is the exception, not the regular route - and it, too, is an addition rather than an intervention.

A real example: SEPA. The first business module taking this route is SEPA payment processing (currently in pilot operation): view mandates, trigger direct debit and credit transfer runs, the archive with repeat download and cancellation, mandate documents from the archive to view, upload and delete. What is remarkable is what did not happen: the business logic - building the file, posting it, archiving it - is the same code that used to run under a Windows interface. What was added is a thin adapter to the REST layer and a description of the screens. Meanwhile both interfaces run side by side on the same tables.



The tiles of the standard and the two of the SEPA module in one row - users cannot tell the difference, and that is exactly the point



The actions area for operations that run something and return a result. All three entries come from the plugin, including description, permission and workflow

8. Tenants, licensing and operations

Tenants. The normal case is your own server in your own house with one tenant - and that is exactly how the system behaves without any further configuration. For hosting, multi-tenant operation can be switched on: one subdomain per customer and one database per customer with its own SQL login, so that SQL Server enforces the separation itself. Device and login tokens are only ever valid under their own tenant, and the active plugins belong to the tenant rather than to the installation.

Licensing. A licence is a signed block bound to the tenant's company data. It carries the number of full workstations (users logged in at the same time) and the additional products enabled. Alongside these there are **app seats** for purpose-built mini clients with their own seat count - a time clock for 500 drivers therefore does not need 500 full workstations.

Expiry locks nobody out. If the licence is missing or has ended, the server switches to **read-only**: all data stays visible, only value-changing posting is blocked - while login, administration and importing a new licence stay open. Add-on modules a customer already licensed in the classic world are carried over automatically when the new licence is issued; nothing is paid for twice.

Administration. Both servers ship with a web interface for operations: first-run assistant and setup diagnostics, users, roles and devices, API clients with scopes and key rotation, migrations, licence status - plus

an evaluation of the access log with key figures, history and the most frequent clients and routes. Remote administration runs over the same routes as everything else, so no session on the server itself is needed.

9. Where it stands, and what comes next

History	Started in January 2025 with the core functions, back then as EulandaXtools. From February 2026 came the foundations for the HTML interface, the WebKit and the module system; the business modules have been built on them since, up to today.
Live	Since September 2026 in daily use at EULANDA itself and set up at the first customers - directly on the EULANDA database, with no synchronisation and no double entry.
Interface	Addresses, articles, quotes, orders, delivery notes, invoices and purchase orders in the browser, on tablet and phone, also as an app on the home screen. Reports as tiles that everyone arranges for themselves. Users, roles and permissions per tenant.
Modules	Customer portal (documents, contracts and requests without a password), ticket system with time tracking, telephony with 3CX, service contracts with recurring invoices, letters, mail merge and newsletters, outgoing e-invoices (ZUGFeRD, XRechnung), EDI with trading partners and SEPA payments.
AI and interfaces	The REST API for any third-party system, an MCP server (Model Context Protocol) through which AI tools work with the data of the suite, and an assistant right in the interface: "Create a quote for customer Mustermann with articles A, B and C" or "Put together the 50 best articles of product group XY and the top customers as an Excel file". It knows the structure of the suite and works directly with its data.
Setup	One setup installs the suite without questions - new, next to an existing EULANDA or as an update. It checks memory, network, firewall and router first, installs SQL Server if needed and gives every company its own address with a valid certificate. After that the suite fetches its updates itself.
Next	Incoming invoices with pre-accounting and transfer to DATEV, then warehouse and stocktaking, dunning, and project, progress and final invoicing - until the suite does everything the classic edition does.
Operations	Parallel operation with the classic edition, plugin system, multi-tenant operation, persistent sessions, licensing and hands-free delivery are in place. Open: load balancing across several servers.

The interface contract is designed to stay free of breaking changes: fields are only added, never renamed or removed, no new mandatory fields are introduced for existing clients, and schema changes come exclusively as new, repeatable migrations. An older application server keeps working against a newer REST server - so updates can be rolled out one after another, without a cut-over date for every site.

10. Technical profile

Database	Microsoft SQL Server holding the existing EULANDA database
Server	Windows, PowerShell 5.1 or newer, http.sys (HttpListener)
Third-party parts	none - no IIS required, no Kestrel, no package jungle
Protocol	REST/JSON, versioned under /api/v1 , errors as problem+json with a stable code
AI	MCP server (Model Context Protocol) for AI tools and an assistant in the interface
Authentication	API key as bearer token, scopes per client, user roles per tenant
Devices	any current browser, Android, iPad, Windows tablet - no installation required
PWA	installation requires HTTPS (or localhost)
Languages	interface and API error texts in English and German, switchable per user
Extensions	plugins with their own schema, routes, permissions and declarative interface - installable per tenant
Setup	one setup for server and database, own address with certificate (such as company.x.eulanda.eu), updates from within the suite
Licensing	signed block per tenant, full workstations and app seats, expiry switches to read-only

EULANDA[®] is a registered trademark of EULANDA Software GmbH. The screenshots show the interface with test data. Full REST API documentation: sdk.eulanda.eu/rest

[EulandaXrest 1.0.5](#) · [EulandaXclient 1.1.3](#) · [As of 25 September 2026](#)