

Entrio

Ein Trading-Journal, das den Screenshot liest und nach der Psychologie fragt, bevor der Trade geschlossen ist

Trader wissen, dass ein Journal hilft, und führen trotzdem keines: Das Erfassen kostet zu viel Zeit. Entrio nimmt einen Screenshot der Handelsplattform, liest die Trade-Daten mit einem Vision-Modell heraus und fragt als einziges Journal am Markt strukturiert nach Emotion, Regeltreue und Setup-Qualität, bevor das Ergebnis feststeht. Dazu kommen vier Broker-Anbindungen in vier Laufzeiten, ein Zahlungssystem, native Clients mit signierter Windows-Distribution und ein Coach-Layer für Trading-Schulen. Konzept, Backend, Frontend, KI-Pipeline, Distribution und Strategie in einer Hand.

rund 124'000

Zeilen eigener Code über vier Repositories,
ohne ExtensionReach

47

Edge Functions in Deno, 78 SQL-
Migrationen, 40 Tabellen

6

Programmiersprachen und Runtimes:
TypeScript, SQL, MQL5, C#, Pascal Script,
Protobuf

360

eigene Commits auf dem Integrationszweig,
ohne Merge- und Generator-Commits

13

externe Dienste im Hauptprodukt, über 20
im Ökosystem

3

Partner-Freigaben: NinjaTrader Vendor,
Rithmic Dev Kit, Spotware Open API

1 Kurzfassung

Entrio ist ein Trading-Journal für diskretionäre Trader. Ein Screenshot der Handelsplattform genügt: Ein Vision-Modell liest Symbol, Richtung, Einstieg, Stop, Ziel, Zeitrahmen und Risiko heraus, danach fragt die App in wenigen Klicks nach Emotion, Regeltreue und Setup-Qualität. Ein KI-Coach und wöchentliche Berichte werten beides gemeinsam aus. Trading-Coaches begleiten ihre Schüler in einem eigenen Bereich.

Der Kern in einem Satz

Die Selbsteinschätzung wird erfasst, bevor der Trade geschlossen ist: Was der Trader über seine Disziplin sagt, ist damit immun gegen den Rückschaufehler, und genau diese Daten hat kein anderes Journal.

Technisch reicht das Projekt von einer React-Anwendung über ein Postgres-Backend mit 102 Row-Level-Security-Policies und 47 serverseitigen Funktionen bis zu Programmen, die in fremden Handelsterminals laufen: ein Expert Advisor in MQL5 und MQL4, ein NinjaTrader-Add-On in C# und WPF, ein von Hand nachgebauter Binärprotokoll-Client für Rithmic. Dazu kommen ein Stripe-Abomodell mit geplanten Downgrades, eine Electron-Tray-App und signierte Windows-Installer aus einer GitHub-Actions-Pipeline. Die interessanten Stellen sind nicht die Breite, sondern die Punkte, an denen es kein Rezept gab: Kapitel 5 nimmt sieben davon auseinander.

2 Ausgangslage und Ziel

Das Problem

Diskretionäre Trader wissen, dass ein Journal die wichtigste Verbesserungsmethode ist. Trotzdem führen es die wenigsten konsequent. Der Grund ist Reibung: Nach einem Trade müssen Symbol, Richtung, Preise, Zeiten und Begründungen von Hand in Excel, Notion oder ein klassisches Journal-Werkzeug übertragen werden. Die etablierten Anbieter lösen das mit Broker-Importen, die laut Nutzerbewertungen ihre häufigste Fehlerquelle sind, und liefern vor allem Auswertungen: was passiert ist. Die eigentliche Frage bleibt offen: warum. Kein etabliertes Journal erfasst strukturiert den emotionalen Zustand, die Regeltreue und die Setup-Qualität pro Trade, obwohl genau diese Faktoren über Erfolg und Scheitern entscheiden.

Das Ziel

Ein Journal, das in zwanzig Sekunden geführt ist und dabei die Daten erhebt, die sonst niemand hat. Konkret in drei Schritten:

- **Erfassen.** Screenshot hochladen, per Datei, Ziehen oder aus der Zwischenablage, bis zu fünf Bilder. Das Vision-Modell erkennt die Plattform, es unterstützt neun davon, und extrahiert alle Felder mit Konfidenzwerten. Alternativ liefern der MetaTrader-Expert-Advisor, das NinjaTrader-Add-On oder die Rithmic-Anbindung dieselben Daten automatisch und broker-verifiziert.
- **Reflektieren, bevor das Ergebnis bekannt ist.** Bei Ein- und Ausstieg fragt Entrio nach Emotionen, Setup- oder Ausführungsqualität A, B oder C, akzeptiertem Risiko, befolgtm Plan und einer Notiz. Auf MetaTrader erscheint diese Abfrage als Desktop-Popup, auf NinjaTrader als natives Panel im Terminal.
- **Lernen.** Ein Dashboard mit 24 Kennzahlen und ein Psychologie-Modul, das den Preis jeder Emotion in Geld ausdrückt, das Verhalten nach Verlusten analysiert und Setup-Qualität gegen Ergebnis stellt. Ein KI-Coach antwortet auf Basis der letzten 200 Trades, wöchentliche und monatliche Berichte kommen per E-Mail und als PDF.

Was Entrio bewusst nicht ist

Kein Social- oder Copy-Trading, kein Broker, keine Signale, kein automatisiertes Trading. Die Produktvision hält zusätzlich fest: Was länger als 30 Sekunden dauert, ist zu kompliziert. Diese Leitplanken sind als Entscheidungen dokumentiert und wurden mehrfach durchgesetzt, etwa beim Verzicht auf eine «Playbooks»-Seite, die als Prototyp gebaut, aber nie ausgeliefert wurde. Auch der Broker-Auto-Import ist bewusst kein Kernversprechen: Die Anbindungen existieren, sind aber Verifikation und nicht Erfassung.

Meine Rolle

- **Strategie und Produkt:** Positionierung, fünf Personas, Preismodell mit Margenrechnung, sechs Architekturentscheidungen mit Querverweisen auf über 40 weitere, 48 datierte Pläne, 21 Research-Dokumente, Partnerverhandlungen bis zum unterzeichneten Vendor-Vertrag.
- **Frontend:** React 18 mit TypeScript, 31 Seiten und 117 Komponenten, sieben Sprachen inklusive Arabisch mit Rechts-nach-links-Layout, Progressive Web App.
- **Backend und Datenbank:** Postgres-Schema mit 40 Tabellen, 102 Row-Level-Security-Policies über fünf Rollenebenen, 47 Edge Functions in Deno, Trigger, Zeitpläne, Rechte-Auflösung als einzelne Datenbankfunktion.
- **KI-Pipeline:** Vision-Prompt mit Plattform-Fingerabdrücken, Schema-Validierung, vier deterministische Nachprüfungen, Kosten- und Audit-Protokoll pro Aufruf, Kontextaufbau für den Coach.
- **Zahlung:** Stripe Checkout, Kundenportal, Webhook mit persistenter Idempotenz, geplante Downgrades über Subscription Schedules, Dunning, E-Mail-Zustellbarkeit.
- **Broker-Anbindungen:** Expert Advisor in MQL5 und MQL4, Add-On in C# und WPF, Rithmic-Binärprotokoll auf Node, MetaApi-Cloud-Sync mit Kostensteuerung.
- **Distribution und Betrieb:** Electron-Tray-App, Inno-Setup-Installer, Signatur über Azure Trusted Signing, GitHub Actions, Admin-Konsole, E-Mail-Infrastruktur, eigene Sicherheitsaudits.

Wie es dazu kam

Der Vorläufer war kein Programm, sondern ein n8n-Workflow: Webhook, Bildanalyse, drei Code-Schritte, Ablage als Notion-Seite. Aus dieser Automatisierung wurde im Januar 2026 eine Anwendung. Gebaut wurde von Januar bis September 2026, rund sieben Monate, davon bis August neben der Berufslehre mit zehn bis zwanzig Wochenstunden und mit einer Entwicklungspause im Juli.

Datum	Schritt
16. Januar 2026	Erster Commit. Ab dem 25. Januar Handarbeit statt Prototyp-Generator: Analyse-Seite, Trade-Service, erste Deno-Funktionen.
17. Februar 2026	Produktions-Go-live von app.entrio.cloud, viereinhalb Wochen nach dem ersten Commit. Davor in wenigen Wochen: KI-Screenshot-Analyse, sieben Sprachen, Emotionserfassung, KI-Coach und Berichte.
20. März 2026	Rithmic gibt ein Developer Kit aus. FTMO antwortet und bestätigt die Abdeckungsstrategie.
27. März 2026	Broker Verification Layer live, Stripe im Live-Modus. Das Produkt kann ab hier Geld entgegennehmen und Trades gegen Broker-Daten prüfen.
19. April 2026	Coach-Kanal validiert: Bei einer Trading-Schule registrieren sich fünf von fünf Schülern in drei Minuten. Der Coach-Layer entsteht aus diesem Pilot.
17. Mai 2026	Vierzehntägiger Trial auf Produktion, 29 Bestandsnutzer werden als eigene Kohorte weitergeführt.
16. Juni 2026	NinjaTrader-Vendor-Vertrag unterzeichnet, Spotware gibt die cTrader Open API frei, und ein selbst durchgeführtes Stripe-Audit findet fünf Mängel im eigenen Bezahlpfad.
3. September 2026	Strategiekorrektur nach eigener Recherche: Coach-Kanal wird Hauptlinie, cTrader und bezahlte Werbung werden gestrichen, die Positionierung auf «Erfassung vor dem Ergebnis» geschärft.

Parallel zum Code liegt seit dem 14. Februar 2026 ein Vault mit 101 Dokumenten: Vision, Personas, Marke, Architekturentscheidungen, Pläne mit Erfolgskriterien und Marktforschung. Die Mission darin lautet wörtlich, Tradern durch Werkzeuge zu helfen, die Selbstreflexion, Disziplin und Lernen fördern. Daraus folgt die Reihenfolge des Bauens: Erst die saubere Erfassung, dann die Auswertung, denn ohne konsistente Emotions- und Strategiedaten steht jede spätere KI-Analyse auf Sand.

3 Funktionsumfang

Drei Rollen, drei Oberflächen, eine Datenbasis.

Trader

- **Erfassen.** Upload mit bis zu fünf Bildern, Einfügen aus der Zwischenablage, Erkennung doppelter Bilder über einen Hash, Emotionsblock für Ein- und Ausstieg, offene Positionen für den späteren Exit, Beispiel-Trade und geführte Tour für den Kaltstart.
- **Journal.** Trade-Liste mit unendlichem Scrollen, Filtern nach Symbol, Status, Richtung und Datum, Mehrfachauswahl, Excel-Export und Live-Aktualisierung, wenn ein Terminal einen Trade liefert. Im Detail-Sheet ist jedes Feld einzeln bearbeitbar, mit Karussell und Lightbox für die Screenshots.
- **Auswerten.** 24 Kennzahlen, Equity-Kurve, Aufschlüsselung nach Symbol, Uhrzeit, Wochentag und Zeitrahmen, dazu ein Psychologie-Radar mit den Kosten jeder Emotion.
- **Lernen.** KI-Coach mit Sitzungsverlauf und Folgefragen, wöchentliche und monatliche Berichte mit PDF-Export.
- **Verbinden.** Kopplung von MetaTrader, NinjaTrader, MetaApi und Rithmic, Notion-Spiegelung, Google-Drive-Sicherung, Chrome Extension, Desktop-App.
- **Konto.** Anmeldung per Einmalcode, Profil, Kontostand, Sprache, E-Mail-Einstellungen, Abo mit Wechsel und Kündigung, Konto löschen.

Coach

- **Einladen ohne Reibung:** ein Code und ein Link mit selbst gewähltem Namen. Wer den Link ohne Konto öffnet, wird nach der Registrierung automatisch verknüpft. Doppel- und Selbstverknüpfung werden abgelehnt.
- **Dashboard:** Schülertabelle mit Anzahl Trades, Gewinnrate, letztem Trade und letzter Aktivität.
- **Schüler-Trades:** Liste unter Row Level Security mit Screenshots, Ein- und Ausstiegsemotionen und einem Notiz-Editor pro Trade. Notizen lassen sich für den Schüler freischalten; er sieht sie in einer Benachrichtigungsglocke mit Direktlink.
- **Aufgaben-Kanban:** Buckets, Aufgaben mit Priorität, Fälligkeit und Anhängen, Zuweisung an Schüler, Abgabepflicht mit Datei-Upload und Abgabe-Review. Gebaut in sechs Tagen für eine Coach-Demo, aktuell auf zwei Konten freigeschaltet.

Betrieb, eigene Konsole

- Nutzerübersicht mit Aktivitäts- und Statusfiltern, Einladen mit Limits und Sprache, Sperren, Löschen inklusive Storage-Bereinigung, Coach-Zuweisung.
- E-Mail-Log mit Suche und Zeitraum, Ankündigungen mit Gelesen-Erfassung, Feedback-Liste, Interview-Hub mit Audio-Upload und automatischer Auswertung.

Die Seiten der Web-App

Gruppe	Seiten, mit Zeilenzahl bei den grossen
Erfassen	<code>Index</code> (840): Ein- und Ausstiegsmodus, bis fünf Bilder, Emotionsblock, offene Positionen, Willkommensdialog, Coach-Banner, Beispiel-Trade.
Journal	<code>Trades</code> (941): Kartenraster mit unendlichem Scrollen, Filter, Mehrfachauswahl, Excel-Export. <code>TradeDetailSheet</code> (978): Karussell, Lightbox, alle Felder inline bearbeitbar, Broker-Datenblock, Coach-Notiz.
Auswerten	<code>Analyse</code> (329) mit 24 Kennzahlen, Equity-Kurve und Aufschlüsselung nach Symbol, Uhrzeit und Wochentag. <code>EmotionAnalysis</code> (668). <code>Insights</code> : Coach-Chat und Berichte mit PDF-Export.
Coach	<code>CoachDashboard</code> (437), <code>CoachStudentTrades</code> (827), <code>CoachTasks</code> (922), <code>StudentTasks</code> (767), <code>JoinCoach</code> (283).
Konto und Zahlung	<code>Settings</code> (1'200) mit Profil, Sprache, Emotions-Schlagworten, Integrationen, Abo und Kontolöschung. Dazu Upgrade in zwei Stufen, Erfolgsseite mit Warten auf den Webhook und Kündigung in drei Schritten mit Halteangebot.
Integrationen	<code>ConnectMetaTrader</code> mit sechsstelliger Codeeingabe, <code>InstallMetaTrader</code> (550) als öffentliche Anleitung mit Versionsmanifest und Hinweis auf die Freigabeliste, <code>ConnectNinjaTrader</code> .
Anmeldung	Login, Registrierung mit Passwortstärke und Sprachwahl, Bestätigung per sechsstelligem Code mit Wartezeit, Passwort vergessen und zurücksetzen.
Betrieb und öffentlich	<code>Admin</code> (1'680). Öffentlich erreichbar: geteilte Trade-Karte über Token, Video-Bibliothek nach Plattform, Fehlerseite.

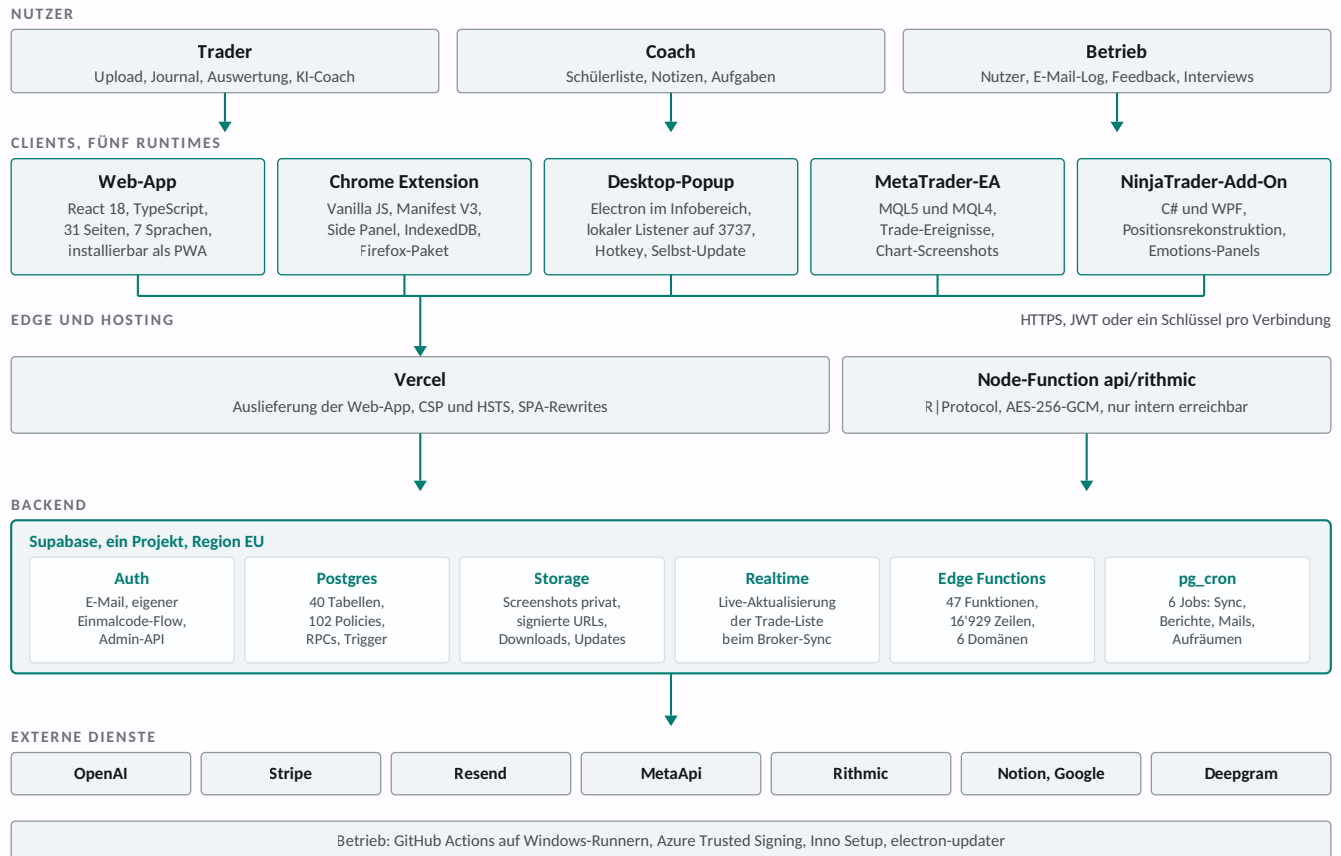
4 Architektur

Eine Serverless-Architektur mit einer Besonderheit: Neben dem Browser liefern vier native Programme Daten in dasselbe Backend, jedes in seiner eigenen Sprache, alle über denselben Kopplungsmechanismus.

Die zehn Schichten

Schicht	Aufgabe und Technologie
1 Web-Frontend	Upload, Journal, Auswertung, Coach-Chat, Einstellungen, Konsole, Progressive Web App. React 18, TypeScript, Vite, Tailwind, shadcn auf Radix, TanStack Query, i18next.
2 Backend und Datenbank	Anmeldung, Datenhaltung, Zugriffsschutz pro Nutzer, Storage, Live-Aktualisierung, 47 serverseitige Funktionen, Zeitpläne. Supabase mit Postgres, Row Level Security, Deno Edge Functions, pg_cron.
3 KI-Pipeline	Screenshot-Extraktion, Exit-Analyse, Coach, Wochen- und Monatsberichte, Interview-Transkription. OpenAI gpt-4.1 im JSON-Modus, Deepgram nova-3, Schema-Validierung mit zod.
4 Zahlung und Rechte	Starter für 15 und Pro für 30 Dollar im Monat, vierzehntägiger Trial ohne Karte, Wechsel in beide Richtungen, Dunning, Nutzungslimits. Stripe Checkout und Webhooks, eine Postgres-Funktion als einzige Wahrheit.
5 Broker-Anbindungen	Automatischer Import und Verifikation aus MetaTrader, NinjaTrader, Rithmic und MetaApi. Expert Advisor in MQL5 und MQL4, Add-On in C# und WPF, R Protocol über WebSocket und Protobuf, Kopplung über Gerätecode.
6 Native Clients	Desktop-Popup im Infobereich, Windows-Installer mit Terminal-Erkennung, signierte Releases, Selbst-Update. Electron, Inno Setup, GitHub Actions, Azure Trusted Signing.
7 Coach-Layer	Dashboard, Einladungslinks, Trade-Notizen, Aufgaben-Kanban mit Abgaben. Eigene Rollenebenen in Row Level Security, eine Edge Function mit 15 Aktionen.
8 Ökosystem	Chrome Extension zur Erfassung aus dem Browser, Marketing-Site, programmatische Videoproduktion, internes Vertriebssystem. Vier eigene Repositories.
9 Qualität und Betrieb	Tests, Sicherheits-Header, Rate Limits, E-Mail-Lebenszyklus, eigene Konsole, Feedback-Kanal. Vitest, Playwright, CSP und HSTS, Resend.
10 Strategie	Vision, Personas, Entscheidungen, Roadmap, Marktforschung, Wettbewerb, Partnerschaften. 101 Dokumente, gepflegt von Februar bis September 2026.

Systembild



Fünf Runtimes liefern in dasselbe Backend. Alle sprechen es über HTTPS an, der Browser mit einem JWT, die Handelsterminals mit einem Schlüssel pro Verbindung. Nur die interne Node-Function auf Vercel spricht das Rithmic-Binärprotokoll.

Datenfluss eines Screenshot-Trades

- Der Client verkleinert das Bild auf 1920 Pixel, komprimiert es, berechnet einen SHA-256-Hash und ruft `upload-and-analyze` mit dem JWT des Nutzers auf.
- Die Funktion prüft JWT, Rate Limit von 20 Anfragen pro Minute und die Rechte des Nutzers, legt die Bilder in einen privaten Bucket und erzeugt Signatur-URLs mit einer Stunde Gültigkeit.
- Alle Bilder gehen in einem einzigen Aufruf an gpt-4.1, im JSON-Modus bei Temperatur 0. Die Antwort wird gegen ein Schema validiert und durch vier deterministische Korrekturen geschickt.
- Geschrieben werden eine Zeile in `trades`, je eine in `trade_media` und eine Audit-Zeile in `trade_parse_results` mit Tokens, Kosten und Laufzeit.
- Danach optional: signierter Webhook nach aussen, Notion-Seite, Google-Drive-Sicherung. Das Frontend bekommt die neue Zeile über Realtime, ohne neu zu laden.

Datenfluss eines Broker-Trades

- Der Expert Advisor läuft im Terminal des Nutzers. Ohne Schlüssel startet er die Kopplung: Er holt einen sechsstelligen Code, zeigt ihn an, der Nutzer bestätigt ihn im Browser, der EA fragt alle fünf Sekunden nach und erhält den Schlüssel genau einmal.
- Beim Öffnen einer Position sendet er Symbol, Richtung, Preise, Kontostand, Tick-Werte und Zeitrahmen. Der Server berechnet Risiko und Chance-Risiko-Verhältnis und antwortet mit einer signierten Upload-URL für den Chart-Screenshot.
- Parallel meldet der EA den Trade an die Desktop-App auf `localhost:3737`. Sie öffnet das Popup und schreibt die Antworten mit demselben Schlüssel zurück.

4. Beim Schliessen liefert er den autoritativen Grund aus `DEAL_REASON`, dazu Gewinn, Kommission und Swap. Der Server klassifiziert das Ergebnis und markiert den Trade als broker-verifiziert.

Vier Grundsätze

- **Ein Rechtesystem für alle.** Die Postgres-Funktion `get_user_entitlements`, inzwischen in achter Fassung, ist die einzige Quelle dafür, was ein Nutzer darf. Frontend und alle 47 Funktionen rufen dieselbe auf.
- **Server entscheidet, Client zeigt an.** Preise werden serverseitig aufgelöst, Limits serverseitig durchgesetzt, Ergebnisse serverseitig klassifiziert. Der Client kann nichts davon umgehen.
- **Idempotenz überall.** Zahlungsereignisse werden protokolliert, E-Mails haben einen Zeitstempel pro Typ, Berichte sind pro Periode eindeutig, Bilder pro Hash, Broker-Tickets pro Verbindung.
- **Ausfallsicher, wo es um Nutzerdaten geht.** Der Signup-Trigger schluckt Fehler, damit eine Registrierung nie scheitert. Der Zahlungs-Webhook antwortet bei Datenbankfehlern mit 500, damit Stripe wiederholt. Cloud-Konten werden im `finally`-Block heruntergefahren, damit keine Kosten entstehen.

Die sechs Domänen der serverseitigen Funktionen

Domäne	Anzahl	Funktionen
KI	6	Screenshot-Analyse als Kern-Pipeline mit 1'629 Zeilen, Exit-Analyse, Trader-Coach, Berichte auf Abruf und nach Zeitplan, Interview-Audio über Deepgram und gpt-4.1.
Zahlung	6	Stripe-Webhook, Checkout, Kundenportal, Plan-Wechsel, Trial-Mails mit 647 Zeilen, Abmeldung von Mails.
Broker	12	Gerätekopplung, Empfang der MetaTrader-Trades mit 880 Zeilen, NinjaTrader-Sitzung und Sync, MetaApi verbinden und synchronisieren, Rithmic verbinden, verifizieren und synchronisieren, Verifikationsstatus.
Anmeldung und E-Mail	12	Einmalcode senden und prüfen, Passwort zurücksetzen, Google-Anmeldung, Warteliste, Einladungs-Backend mit 766 Zeilen, Konto löschen, Adminprüfung, Berichtsversand.
Coach	2	Beitritt über Einladung, Aufgabenverwaltung mit 15 Aktionen in einer Funktion.
Integrationen	9	Sieben Notion-Funktionen von OAuth bis Sync, Google-Drive-Sicherung, signierter Bildproxy, Lead-Suche im eigenen Vertriebssystem.

44 der 47 Funktionen laufen ohne JWT-Prüfung am Gateway und prüfen selbst, damit abgelaufene Tokens sauber behandelt werden können. Ein Test liest die Konfiguration und stellt sicher, dass jede dieser Funktionen eine eigene Prüfung enthält.

Die vier Broker-Anbindungen

Anbindung	Technik	Was sie leistet	Stand
MetaTrader 5 und 4	MQL5 und MQL4, 984 und 765 Zeilen	Läuft im Terminal. Kopplung, Ereignisse bei Öffnen, Schliessen und Änderung von Stop und Ziel, Chart-Screenshots über signierte Upload-URL, Meldung an die Desktop-App, Selbstprüfung der Version, Zeitzonekorrektur, keine DLL-Importe.	MT5 produktionsreif , MT4 bewusst nicht ausgeliefert
NinjaTrader 8	C#, .NET 4.8, WPF, 2'373 Zeilen	Abonniert die Ausführungen aller Konten, rekonstruiert Positionen mit Aufstocken, Abbauen und Drehen, liest Bracket-Orders für Stop und Ziel, macht Chart-Screenshots über die offizielle Schnittstelle, zeigt eigene Panels für die Emotionserfassung.	fertig seit April 2026, ein Live-Test offen
Rithmic	Node auf Vercel, WebSocket und Protobuf, 589 Zeilen	Von Hand gebauter R Protocol-Client: Anmeldung am Order-Plant, Kontenliste, Fill-Historie nach Handelstag, Demultiplexen über Template-IDs, mehrteilige Antworten. Zugangsdaten verschlüsselt, nur intern erreichbar.	Testumgebung grün, Produktion wartet auf das Konformitätsverfahren
MetaApi	Deno Edge Functions, 1'645 Zeilen	Richtet ein Cloud-Konto mit Investor-Passwort ein, das nie gespeichert wird, holt die Deal-Historie mit Rückfallebene, ordnet Trades über ein Toleranz-Scoring zu und synchronisiert täglich zur gewählten Uhrzeit.	live seit März 2026

Vier Schwester-Repositories

- **Chrome Extension.** Trade-Erfassung aus jedem Tab, Manifest V3 ohne Bundler, eigener REST-Client statt SDK, weil dieses im Service Worker nicht sauber läuft. Entwürfe in IndexedDB, Upload überlebt das Schliessen des Fensters, Paket auch für Firefox.
- **Homepage.** Eigenes Frontend-Projekt mit 21 Seiten, 28 Landing-Bausteinen und 30 Animationskomponenten. Jede inhaltliche Seite hat einen deutschen und einen englischen Pfad statt eines Sprachparameters, damit jede Sprache eine indexierbare Adresse hat.
- **Remotion.** Videos als Code: 14 Kompositionen und 16 wiederverwendbare Szenen. Die Plattform-Anleitungen werden aus einer Liste von Plattform-Sprache-Paaren erzeugt, eine neue Sprache ist ein Eintrag und kein neues Video.
- **ExtensionReach.** Internes Vertriebssystem, das Lead-Suche, Pipeline, Entwürfe und Antwort-Klassifikation zusammenführt, mit zwei eigenen Chrome Extensions. Der Admin-Interview-Hub in Entrio sucht Leads direkt in dessen Datenbank. Dazu liegt eine eigene Projektdokumentation vor, sie ist die massgebliche Quelle für die Zahlen zu diesem Projekt.

Entscheidungen, die den Bau geprägt haben

Nummer	Datum	Entscheidung
001	Februar 2026	Supabase als Backend für alle Projekte, weil es Postgres mit Row Level Security, Auth, Storage, Deno-Funktionen und eine REST-API bündelt, die auch aus einer Chrome Extension erreichbar ist. Die Bindung an einen Anbieter wird durch die Portabilität von Postgres gemildert.
035	22. März 2026	UX-Audit gegen Hick, Fitts, Miller und Jakob: Bottom-Navigation von fünf auf vier Ziele, linearer Upload, gruppierte Einstellungen.
038	27. März 2026	Broker Verification Layer live, zuerst über MetaApi und Rithmic.
040	4. April 2026	Die Entwicklungsumgebung wird aufgegeben, es gibt nur noch Produktion. Begründung: Drift zwischen zwei Umgebungen kostet mehr, als die Trennung bringt.
041	14. August 2026	Das Warnsymbol folgt dem Prüfstatus, nicht der Anzahl Warnungen; eine rechnerisch ergänzte Kennzahl ist keine Warnung. Datenbasis: 35 von 59 Markierungen waren Rauschen. Prinzip: lieber null als falsch.
043	Mai 2026	Das Desktop-Popup bleibt auf MetaTrader beschränkt, weil NinjaTrader eigene Panels im Terminal hat. Kein zweiter Weg für dasselbe Ziel.

5 Technische Kernprobleme

Sieben Stellen, an denen es kein Rezept gab. Jede Karte nennt Problem, Lösung und das Prinzip dahinter.

AUTHENTIFIZIERUNG

Anmeldung für Programme ohne Browser

Problem

Der Expert Advisor läuft im MetaTrader-Terminal, das Add-On in NinjaTrader. Beide haben keinen Browser, keine OAuth-Bibliothek und keinen Ort, an dem ein Nutzer sicher ein Passwort eingeben könnte. Ein Zugangsschlüssel in einer Konfigurationsdatei wäre die einfache Lösung und läge damit im Klartext auf jedem Rechner, auf dem das Terminal läuft.

Lösung

Ein Ablauf nach dem Muster der OAuth Device Authorization, zweimal umgesetzt. Das Terminal holt ohne Anmeldung einen kurzen Nutzercode und einen geheimen Gerätecode, Gültigkeit zehn Minuten. Der Nutzer bestätigt den kurzen Code in der eingeloggten Web-App. Das Terminal fragt alle zwei bis fünf Sekunden nach und erhält den Schlüssel **genau einmal**, danach wird die Sitzung gelöscht. Ein abgelaufener Versuch bekommt 410 und keine Wiederholung. Die Desktop-App hat gar keine eigenen Zugangsdaten: Der EA reicht Schlüssel und Trade-Kennung im lokalen Ereignis mit.

Warum es zählt

Ein Anmeldeverfahren wird von den Fähigkeiten des schwächsten Beteiligten bestimmt, nicht von denen des Servers. Wo kein Browser ist, muss die Vertrauenskette über ein zweites, bereits vertrauenswürdiges Gerät laufen, und das Geheimnis darf die Strecke nur ein einziges Mal passieren. Wer stattdessen den Schlüssel dauerhaft ablegt, verschiebt das Problem nur an einen Ort, an dem es niemand mehr sieht.

MODELLGRENZEN

Was das Sprachmodell nicht entscheiden darf

Problem

Ein Vision-Modell liest aus einem Chart-Screenshot Zahlen heraus und ist dabei überzeugend, auch wenn es sich irrt. Eine vertauschte Richtung oder ein Zahlendreher im Einstieg macht jede spätere Auswertung falsch, ohne dass irgendwo ein Fehler auftaucht.

Lösung

Der Prompt beschreibt für jede der neun Plattformen, woran sie zu erkennen ist und wo die verlässlichsten Werte stehen. Danach greifen vier deterministische Prüfungen: Die Richtung folgt der Preisgeometrie, denn liegt der Stop über dem Einstieg, ist es ein Verkauf, unabhängig davon, was das Modell sagt. Zahlen aus der Freitext-Notiz werden gegen die strukturierte Antwort gestellt und im Zweifel bevorzugt. Das Chance-Risiko-Verhältnis wird nachgerechnet statt übernommen. Am Ende steht ein Prüfstatus. Die Schwelle dafür wurde im August an 59 tatsächlich markierten Trades kalibriert: 35 davon waren nur wegen einer rechnerischen Ergänzung markiert, also Rauschen. Seither ist Rechnen keine Warnung mehr.

Warum es zählt

Ein Modell liefert Vorschläge, keine Fakten. Überall dort, wo eine Regel die Antwort eindeutig bestimmt, gehört die Regel in den Code und nicht in den Prompt, denn sie ist prüfbar und das Modell ist es nicht. Und eine Warnung, die zu oft grundlos erscheint, erzieht den Nutzer dazu, sie zu ignorieren: Ihre Schwelle ist deshalb eine Messgrösse und keine Geschmacksfrage.

PROTOKOLL

Ein Binärprotokoll ohne Bibliothek

Problem

Rithmic liefert die Fills für Futures-Konten über R|Protocol, eine WebSocket-Verbindung mit Protobuf-Nachrichten. Es gibt dafür kein JavaScript-Paket, nur 16 Schema-Dateien und eine Beschreibung. Auf derselben Verbindung laufen alle Nachrichtentypen gemischt, und eine Antwort kann über mehrere Frames verteilt sein.

Lösung

Ein eigener Client mit 589 Zeilen auf einer Node-Function: Anmeldung am Order-Plant, Kontenliste, Fill-Historie nach Handelstag. Jede Nachricht trägt eine Template-ID, über die der Client sie dem richtigen Verarbeiter zuordnet; mehrteilige Antworten werden gesammelt, bis der Abschluss kommt. Die Zugangsdaten liegen AES-256-GCM-verschlüsselt in der Datenbank, entschlüsselt wird nur in dieser Function. Weil dieselben Daten aus Deno und aus Node gelesen werden, ist das Wire-Format bewusst identisch gehalten.

Warum es zählt

Eine fehlende Bibliothek ist kein Ausschlusskriterium, sondern eine Aufwandsfrage. Ein Protokoll ist eine Spezifikation, und wer sie liest, kann sie umsetzen. Der eigentliche Gewinn liegt darin, dass man danach genau weiss, was über die Leitung geht, statt es einer fremden Abstraktion zu glauben. Der Preis ist ebenso klar: Man erbt die Pflege, und deshalb gehört so ein Client hinter eine schmale Grenze und nicht in die Anwendung.

ZAHLUNG

Ereignisse, die in falscher Reihenfolge kommen

Problem

Stripe garantiert weder die Reihenfolge der Webhook-Ereignisse noch, dass jedes genau einmal ankommt. Wer den Zustand aus dem Ereignis übernimmt, schreibt irgendwann einen alten Stand über einen neuen und macht einen zahlenden Kunden zum abgelaufenen Trial.

Lösung

Bei jedem Abo-Ereignis wird die Subscription frisch von Stripe geladen, statt dem Inhalt des Ereignisses zu glauben. Die Ereignis-ID wird in einer Tabelle festgehalten, aber erst nach erfolgreicher Verarbeitung, damit ein Fehlschlag wiederholt werden kann; ein Datenbankfehler beantwortet den Webhook bewusst mit 500. Die Zuordnung zum Nutzer läuft über drei Stufen. Ein Downgrade wird nicht sofort ausgeführt, sondern als zweiphasiger Subscription Schedule auf das Periodenende gelegt, dessen Kennung gespeichert und widerrufen werden kann. Zwei selbst durchgeführte Audits im Juni und August fanden zehn Mängel mit Datei und Zeile, darunter einen Tippfehler in einer Preis-ID, der jeden Starter-Kauf hätte scheitern lassen.

Warum es zählt

In einem verteilten System ist die Nachricht ein Hinweis, dass sich etwas geändert hat, und nicht der neue Zustand. Wer sie als Zustand behandelt, baut eine Anwendung, die von der Laufzeit des Netzes abhängt. Und beim Geld reicht Testen nicht: Fehler treten dort selten und teuer auf, deshalb ist ein Audit mit Fundstellen der einzige Weg, der die eigene blinde Stelle findet.

PLATTFORMUNTERSCHIED

Warum es die MetaTrader-4-Fassung nicht in den Versand schafft

Problem

Der Expert Advisor existiert zweimal, in MQL5 mit 984 und in MQL4 mit 765 Zeilen. MQL5 kennt Positionen und Deals und meldet jede Veränderung über ein Ereignis. MQL4 kennt nur eine Orderliste und hat kein solches Ereignis. Schwerer wiegt: MQL4 hat kein Gegenstück zu `DEAL_REASON`, kann also nicht sagen, ob ein Trade durch Ziel, Stop, Stop-Out oder von Hand geschlossen wurde.

Lösung

Die MT4-Fassung lebt allein von einem Zeitgeber: Sie läuft alle zwei Sekunden über die Orderliste, vergleicht sie gegen eine gemerkte Ticket-Liste und leitet Öffnen und Schliessen aus der Differenz ab. Kopplung, Screenshot-Upload, Desktop-Meldung und Installer sind identisch. Den Schliessgrund liefert sie nicht, und der Server rät ihn auch nicht aus Preisen.

Deshalb wird die Fassung nicht ausgeliefert, obwohl sie fertig ist.

Warum es zählt

Zwei Plattformen mit demselben Namen können unterschiedliche Dinge über die Welt wissen, und dieser Unterschied lässt sich nicht wegprogrammieren. Die ehrliche Antwort darauf ist, die Grenze zu benennen und den schwächeren Weg zurückzuhalten. Die bequeme Alternative, die fehlende Angabe zu schätzen und das Ergebnis wie eine Messung aussehen zu lassen, beschädigt genau die Daten, für die es das Produkt gibt.

ZUSTANDSREKONSTRUKTION

Aus einzelnen Ausführungen wieder einen Trade machen

Problem

NinjaTrader meldet keine Trades, sondern Ausführungen. Ein Trader stockt auf, baut teilweise ab, dreht die Richtung mitten in der Position. Aus diesem Strom muss serverseitig hervorgehen, was ein Trader als einen Trade versteht, samt Durchschnittspreisen und realisiertem Ergebnis. Zusätzlich liefert die Plattform Stop und Ziel nicht am Trade, und der Weg zum Chart-Screenshot ist nicht dokumentiert.

Lösung

Ein Zustandsspeicher pro Konto und Instrument. Eintritts-Ausführungen öffnen oder stocken auf, mit volumengewichtetem Durchschnitt; Austritts-Ausführungen bauen ab; Menge null schliesst mit Durchschnittspreisen und realisiertem Ergebnis; eine entgegengesetzte Ausführung löst eine Drehung aus. Jede Position bekommt eine Kennung, die Öffnen und Schliessen serverseitig verbindet. Stop und Ziel kommen aus den arbeitenden Bracket-Orders desselben Instruments. Der Screenshot läuft über die offizielle Schnittstelle, deren Aufruf über Reflection gefunden wird, inklusive Abgleich der unterschiedlich geschriebenen Kontraktnamen. Wenn nach einem Neustart das Öffnen-Ereignis fehlt, legt der Server den Trade als vollständigen geschlossenen Trade an.

Warum es zählt

Fremde Systeme geben ihre Daten in der Form heraus, die zu ihrem eigenen Modell passt, und fast nie in der, die die eigene Domäne braucht. Diese Übersetzung ist der Kern der Integration und nicht ihr Randfall; sie gehört an eine Stelle, die man für sich prüfen kann. Und weil ein Prozess abstürzen, neu starten und Ereignisse verpassen kann, muss der Empfänger mit Lücken umgehen können, statt einen lückenlosen Strom vorauszusetzen.

DISTRIBUTION

Eine Windows-Anwendung, der das System vertraut

Problem

Eine unsignierte Installationsdatei wird von Windows als Risiko gemeldet, und die Zielgruppe sind Trader, die auf demselben Rechner ihr Handelskonto führen. Dazu kommt der häufigste Supportfall überhaupt: MetaTrader blockiert Netzwerkzugriffe eines Expert Advisors, solange die Adresse nicht in einer Freigabeliste in einer Konfigurationsdatei des Terminals steht.

Lösung

Eine Pipeline, die an einem Git-Tag hängt: Build, Signatur über Azure Trusted Signing mit SHA-256 und einem Zeitstempel nach RFC 3161, danach Neuberechnung von Prüfsumme und Grösse im Update-Manifest, weil die Signatur die Bytes verändert und der Selbst-Updater sonst jede Aktualisierung ablehnt. Der Installer, vier Pascal-Skripte mit 558 Zeilen, sucht die Terminals im Benutzerprofil, erkennt Version 4 und 5 an ihren Ordnern, kopiert die kompilierten Programme hinein und trägt die drei benötigten Adressen doppelungsfrei in die Freigabeliste ein. Keine Administratorrechte nötig.

Warum es zählt

Software endet nicht am erfolgreichen Build. Alles zwischen dem fertigen Programm und dem laufenden Programm auf einem fremden Rechner ist Teil des Produkts: Vertrauensketten, Update-Manifeste, fremde Konfigurationsdateien. Wer diesen Weg dem Nutzer überlässt, gibt eine Anleitung heraus, die die Hälfte nicht zu Ende liest, und misst danach die eigene Aktivierungsrate.

6 Datenmodell

40 Tabellen in Postgres, 28 davon mit Row Level Security und zusammen 102 Policies. Alles hängt an der Nutzererkennung aus der Anmeldung.

Das Herz ist `trades` mit rund 45 Spalten in sechs Gruppen: Identität und Quelle, also Screenshot, MetaTrader oder NinjaTrader; Setup mit Symbol, Richtung, Zeitrahmen, Einstieg, Stop, einem oder mehreren Zielen, Chance-Risiko-Verhältnis und Risiko in Prozent; Ergebnis mit Ausstieg, Status als geprüfter Wertebereich, Gewinn absolut und in Prozent, Kommission und Swap; Qualität mit Konfidenz pro Feldgruppe, Warnungen und Prüfstatus; Psychologie als zwei JSONB-Felder für Ein- und Ausstieg; Verifikation mit Rohdaten des Brokers, Ticket, Verbindung und Kontostand bei Eröffnung. Daneben stehen `trade_media` für die Screenshots, mit einem eindeutigen Index auf Nutzer und Bild-Hash, und `trade_parse_results` als Audit-Spur mit Modell, Tokens, Kosten in Dollar, Laufzeit und Wiederholungen.

Gruppe	Tabellen
Journal	<code>trades</code> , <code>trade_media</code> , <code>trade_parse_results</code> , <code>trade_uploads</code> , <code>user_emotion_tags</code> , <code>user_strategies</code> , <code>strategy_rules</code> , <code>strategy_media</code> , <code>trade_compliance</code>
Nutzer und Zugang	<code>user_settings</code> , <code>user_roles</code> mit den Rollen Admin, Moderator, Nutzer, Beta-Tester und Coach, <code>user_verification</code> , <code>email_otp_codes</code> , <code>beta_access</code> , <code>waitlist</code> , <code>feedback</code> , <code>announcements</code>
Zahlung und Betrieb	<code>subscriptions</code> mit Trial-Zeitstempeln, Tier, Kohorte, sechs Idempotenz-Spalten für Mails und geplantem Plan-Wechsel, <code>usage_limits</code> , <code>stripe_events</code> , <code>rate_limit_log</code> , <code>email_requests</code>
Integrationen	<code>user_integrations</code> für OAuth-Tokens, <code>mt_connections</code> für alle vier Broker-Wege mit verschlüsselten Rithmic-Zugangsdaten, <code>ea_device_sessions</code> , <code>connect_sessions</code>
KI	<code>ai_reports</code> , eindeutig pro Nutzer, Typ und Periode, <code>chat_sessions</code> , <code>chat_messages</code>
Coach	<code>coach_invite_codes</code> , <code>coach_students</code> , <code>coach_trade_notes</code> und fünf Tabellen für Aufgaben, Zuweisungen, Anhänge und Abgaben
Forschung	<code>interview_contacts</code> , <code>interview_entries</code> mit Transkript, Stimmung, Schlagworten und Verweis auf den Lead im Vertriebssystem
Storage	Fünf Buckets: Screenshots privat mit signierten URLs, Coach-Dateien, Abgaben, Installer, Desktop-Updates

Zugriffsschutz in fünf Ebenen

- **Pro Nutzer:** Die eigene Kennung muss mit der Zeile übereinstimmen, auf allen Tabellen mit Nutzerdaten.
- **Systemrolle:** ausdrückliche Regeln auf Systemtabellen. Zahlungsereignisse und Kopplungssitzungen sind für Clients vollständig gesperrt.
- **Betrieb:** eine Hilfsfunktion mit erhöhten Rechten prüft die Rolle, verwendet auf E-Mail-Log, Interviews und Einladungscodes.
- **Coach und Schüler:** Der Coach besitzt Buckets und Aufgaben. Schüler lesen Aufgaben nur über eine bestehende Zuweisung. Trade-Notizen sieht ein Schüler nur, wenn der Coach sie freigibt. Das Anlegen setzt eine bestehende Verknüpfung voraus.
- **Storage:** Trennung über den Pfad, das erste Ordnersegment muss die eigene Nutzererkennung sein.

Dazu kommen acht Trigger und über 20 Funktionen mit erhöhten Rechten: Der Signup-Trigger legt Trial und Einstellungen an und schluckt dabei Fehler, damit eine Registrierung nie scheitert. Ein Trade-Trigger normalisiert die Richtung und berechnet das

Chance-Risiko-Verhältnis vor jedem Schreiben. Ein Notiz-Trigger setzt den Gesehen-Status nur zurück, wenn sich der Text wirklich ändert. Sechs Zeitpläne laufen mit bewusst versetzten Minuten, damit die Last sich verteilt: Aufräumen zur vollen Stunde, MetaApi-Sync ebenfalls, Rithmic fünf Minuten später, Trial-Mails nach fünfzehn, Wochenberichte montags und Monatsberichte am Ersten.

Eine Härtungsrunde, die ein echtes Leck geschlossen hat

Im August 2026 wurden alle Funktionen mit erhöhten Rechten durchgegangen: Ausführungsrechte für nicht angemeldete und für alle Rollen entzogen, Aufrufer-Prüfungen ergänzt und eine Statistikfunktion nach einem nachgewiesenen Datenleck auf Coach und Betrieb beschränkt. Ein Test in der End-zu-Ende-Suite prüft seither, dass fremde Trades tatsächlich unsichtbar bleiben.

7 Qualität und offene Punkte

Getestet wird dort, wo ein Fehler still bleibt: an den Verträgen zwischen Client und Server, nicht an der Oberfläche.

Massnahme	Was sie leistet
Unit-Tests	17 Dateien mit rund 314 Fällen. Schwerpunkt sind die 25 Statistikfunktionen mit 75 Fällen, dazu Trade-Service, Chat und Berichte, Zahlung, Hooks mit gestellten Zeitgebern, Passwortstärke und die Übereinstimmung der Übersetzungen zwischen Deutsch und Englisch.
Konfigurations-Tests	Ein Test liest die Deployment-Konfiguration und prüft Content-Security-Policy und HSTS; ein zweiter liest die Funktionskonfiguration und stellt sicher, dass jede Funktion ohne JWT-Prüfung am Gateway eine eigene Prüfung enthält. Ein dritter verhindert, dass Stripe-Preis-Kennungen ins ausgelieferte Bundle gelangen.
End-zu-Ende-Tests	14 Spezifikationen, 89 Fälle, jeder auf drei Bildschirmgrößen, zusammen 267 Läufe. Die Hälfte prüft API-Verträge direkt: Isolation zwischen Nutzern, SQL-Injection in Abfrageparametern, Format der Rate-Limit-Antworten, dass fehlerhafte Nutzlast nie einen Serverfehler ergibt, der vollständige Lebenszyklus eines Trades.
Eigene Audits	Der Bezahlpfad wurde im Juni und im August je einmal systematisch durchgesehen, mit Datei- und Zeilenangaben. Im Juni wurde zusätzlich die gesamte Angriffsfläche auditiert und eine priorisierte Massnahmenliste erstellt.
Härtung im Transport	HSTS über zwei Jahre mit Preload, Content-Security-Policy mit ausgehenden Verbindungen nur zum eigenen Backend, Frames verboten, Kamera, Mikrofon und Ort abgeschaltet, Herkunftsprüfung mit fester Liste statt Platzhalter, Prüfung der Rücksprungadressen beim Checkout gegen Open Redirect.
Rate Limits	Gleitendes Fenster in der Datenbank, atomar ausgewertet, mit Angabe der Wartezeit in der Antwort. Tier-abhängig in den KI-Endpunkten, drei Einmalcodes pro Adresse in zehn Minuten, 60 Terminal-Anfragen pro Minute.

Betriebsvorfälle, gefunden und behoben

- Zwei Berichts-Zeitpläne scheiterten seit Juni still an einer Typumwandlung in JSONB. Sie liefen, meldeten nichts und schrieben nichts.
- Ein Commit im April entfernte die Willkommensmail für Selbstregistrierte. Elf Betroffene, Versand nachgeholt.
- Ein Registrierungsformular verlor den Namen zwischen Eingabe und Speicherung.

- Die Live-Aktualisierung der Trade-Liste brauchte eine Einstellung in der Datenbank, die im Code nicht sichtbar ist.

Bewusste Grenzen des aktuellen Stands

- **Keine Test-Pipeline.** Unit-, Konfigurations- und End-zu-Ende-Tests laufen lokal. Die beiden vorhandenen Automatisierungen sind Release-Pipelines für Desktop-App und Installer.
- **TypeScript nicht im strikten Modus,** 82 untypisierte Stellen ausserhalb von Oberfläche und Tests.
- **Sechs Dateien über 900 Zeilen.** Fünf davon sind Seiten, die in Kapitel 3 mit derselben Zeilenzahl aufgeführt sind; die grösste, das Integrations-Sheet mit 2'020 Zeilen, ist eine Komponente und taucht dort nicht auf. Und 17 Seiten holen ihre Daten noch von Hand statt über die Query-Schicht, die überall sonst verwendet wird.
- **Toter Code im Bundle:** der nie ausgelieferte Playbooks-Prototyp, ein Debug-Werkzeug, zwei ungenutzte Abhängigkeiten.
- **Offene Sicherheitspunkte aus dem Juni-Audit:** Rotation des Zeitplan-Secrets, unverschlüsselte OAuth-Tokens für Notion und Google, zwei Zeitplan-Funktionen ohne eigene Prüfung, abgesichert nur durch eine Zeitsperre und Kostengrenzen, und öffentlich lesbare Coach-Buckets.

Stand des Repositories

Das Projekt hat zwei Hauptlinien. Die Produktionslinie wurde im August durch gezielte Übernahmen neu aufgebaut und lässt Desktop-App, Installer, Add-On-Quelle und die Release-Pipelines bewusst aussen vor. Die vollständige Historie liegt auf einem Integrationszweig, der 43 Commits voraus ist und als einziger echte, nicht anderweitig übernommene Arbeit enthält. Dazu kommt eine noch nicht eingeecheckte Härtung des Bezahlpfads über neun Funktionen, die den Unterschied zwischen den beiden Bezahlstufen erstmals tatsächlich durchsetzt. Der Zusammenführung dieser Linien steht nichts entgegen; sie ist der erste Punkt im Plan vom 3. September.

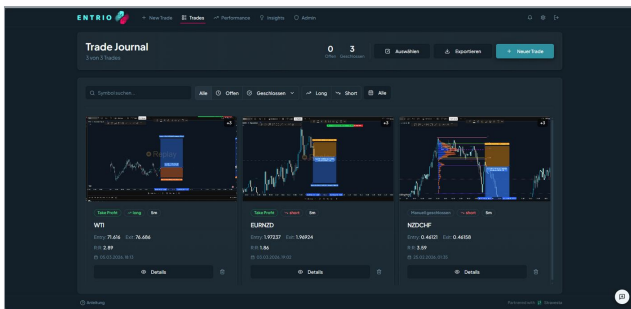
Was die Nutzung zeigt

Im August 2026 kamen 62 von 62 geprüften Trades über einen Screenshot herein, keiner über den Expert Advisor, das Add-On oder Rithmic. Das ist der wichtigste Befund des Sommers und im Plan vom 27. August offen benannt: Die Screenshot-Erfassung ist der validierte Kern, die Broker-Anbindungen sind technisch fertig, aber noch nicht nachgefragt.

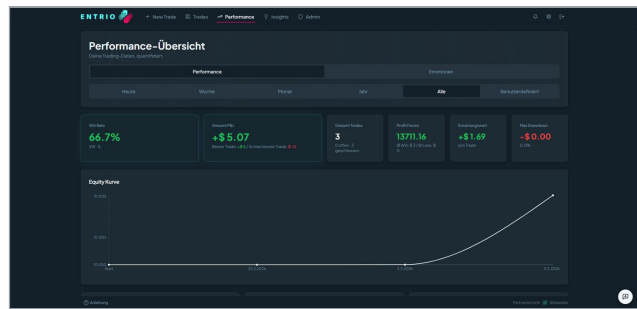
Einordnung

Entrio ist technisch weit fortgeschritten und steht vor dem kommerziellen Beweis. Es hat aktive Nutzer, aber noch keinen zahlenden Kunden. Das Verhältnis von Bauleistung zu Vertriebsergebnis ist der zentrale Lernpunkt des Projekts, und er ist selbst diagnostiziert: «Der Bottleneck ist Distribution, nicht Produkt» steht im April 2026 in den eigenen Notizen, und im September der Satz, an einem Tag seien drei Recherchen gefahren und null Mails verschickt worden. Die Konsequenz ist ein enger Plan: Bezahlpfad schliessen, Coach-Kanal, die letzten Meter bei NinjaTrader, in Tagen statt in Wochen.

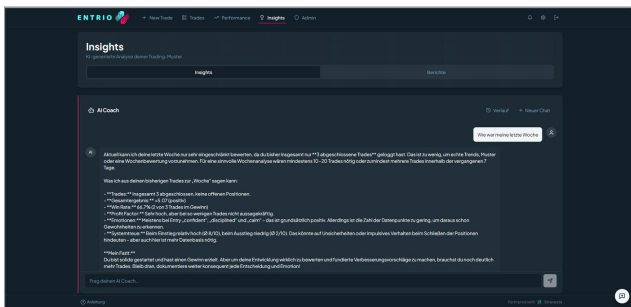
8 Einblicke



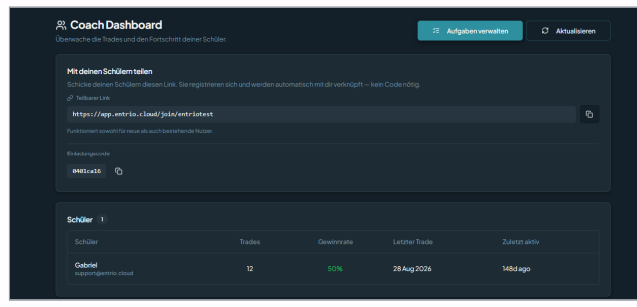
Journal: Screenshot-Karten mit Status, Richtung, Zeitrahmen, Ein- und Ausstieg. Filter nach Symbol, Status, Richtung und Datum.



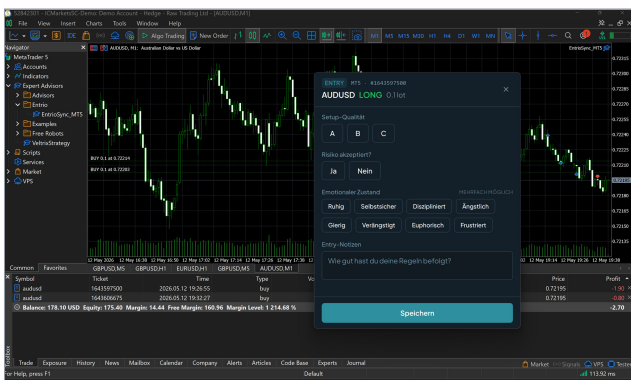
Auswertung: Kennzahlen und Equity-Kurve, umschaltbar zwischen Leistung und Psychologie.



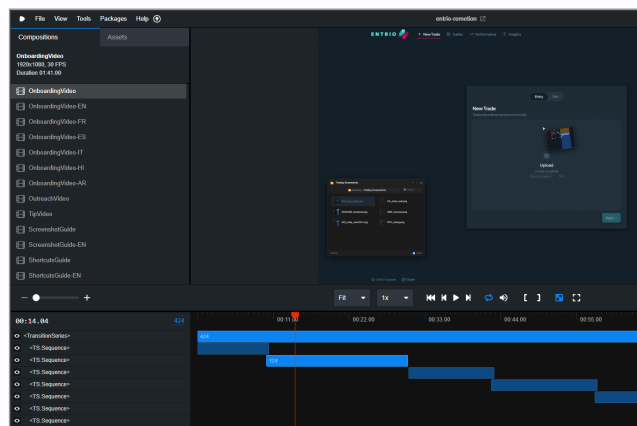
KI-Coach mit Sitzungsverlauf. Jede Anfrage bekommt einen statistischen Kontext aus den letzten 200 Trades, inklusive Emotionshäufigkeiten mit Trefferquoten.



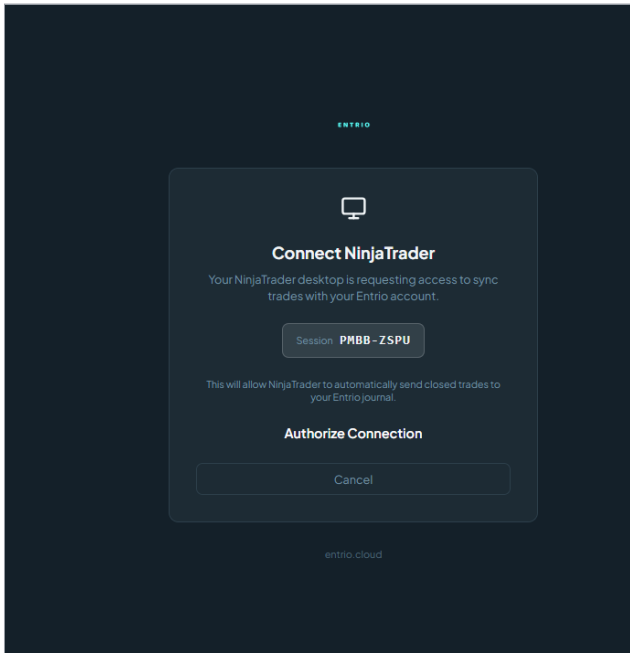
Coach-Dashboard: Einladungslink mit selbst gewähltem Namen, darunter die Schülertabelle mit Trades, Gewinnrate und letzter Aktivität.



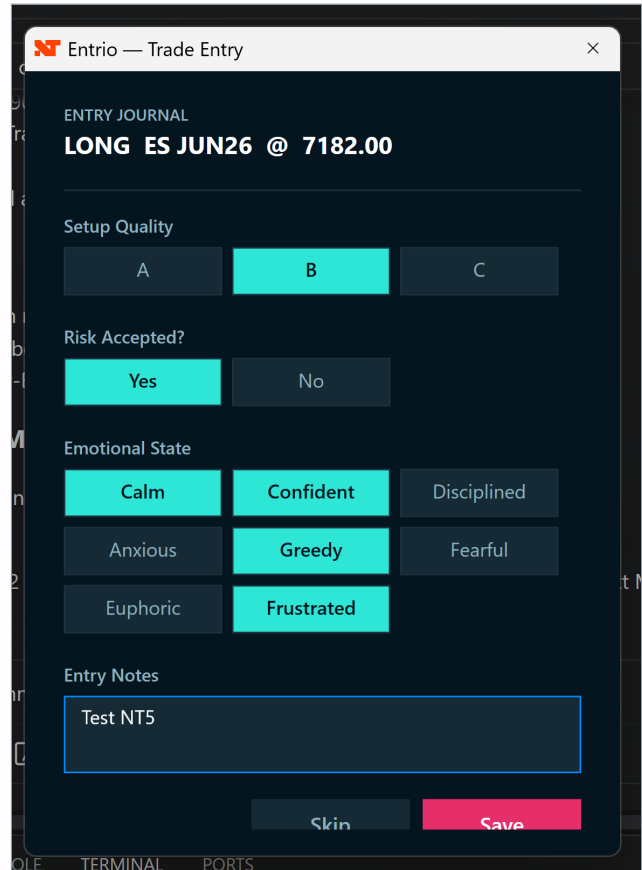
Das Popup über dem Handelsterminal: Der Expert Advisor meldet den Trade lokal, die Desktop-App fragt nach Qualität, Risiko, Emotion und Notiz, bevor das Ergebnis feststeht.



Videos als Code: Onboarding in sieben Sprachen, Anleitungen je Plattform, erzeugt aus einer Liste von Plattform-Sprache-Paaren statt einzeln geschnitten.



Kopplung: Das Terminal zeigt einen kurzen Code, der Nutzer bestätigt ihn im eingeloggten Browser. Danach holt sich das Terminal seinen Schlüssel genau einmal ab.



Erfassung im Handelsterminal selbst: zwei WPF-Fenster ohne Beschreibungsdatei, vollständig aus C#-Code aufgebaut, mit denselben Feldern wie im Web.

9 Was das Projekt zeigt

Nachgewiesen heisst hier: im Repository vorhanden, in Produktion oder in einem funktionierenden Build, nicht nur geplant.

Technisches Handwerk

Bereich	Stand	Beleg im Projekt
React und TypeScript	Produktion, 42'947 Zeilen	Hooks-Architektur, TanStack Query, zod, Recharts, Drag-and-Drop, PDF-Export, sieben Sprachen mit Rechts-nach-links-Layout, Progressive Web App, mobile Safe-Areas.
Serverless-Backend	Produktion, 16'929 Zeilen	47 Funktionen in Deno, sechs Authentifizierungsmuster, Rate Limiting, Idempotenz, Webhook-Verarbeitung, OAuth-Clients für Google und Notion, E-Mail-Zustellbarkeit.
PostgreSQL	Produktion, 78 Migrationen	Schema mit JSONB, geprüften Wertebereichen und eindeutigen Indizes, 102 Policies über fünf Rollenebenen, gehärtete Funktionen mit erhöhten Rechten, Trigger, Zeitpläne, Storage-Regeln.
Arbeiten mit Sprachmodellen	Produktion	Vision-Prompt mit Plattform-Fingerabdrücken, JSON-Modus, Schema-Validierung, vier deterministische Nachprüfungen, Kostenprotokoll pro Aufruf, Kontextaufbau aus 200 Trades, datengetriebene Kalibrierung der Warnschwelle.
Zahlung und Abomodell	Produktion, live seit März	Checkout, Kundenportal, Webhook mit persistenter Idempotenz, Ereignisse in falscher Reihenfolge, geplante Downgrades, Dunning, Kohorten-Zuordnung, Gutscheincodes.
MQL5 und MQL4	Gebaut, MT5 produktionsreif	1'749 Zeilen in zwei Expert Advisors gegen zwei verschieden gebaute Schnittstellen, Netzzugriff ohne DLL, Kopplung, Screenshot-Upload, Zeitzonen, Selbstprüfung der Version, von Hand geschriebene JSON-Behandlung.
C#, .NET und WPF	Gebaut, ein Live-Test offen	2'373 Zeilen: Ausführungsereignisse, Positionsrekonstruktion mit Drehung, Reflection gegen Interna der Plattform, Screenshot-Schnittstelle, im Code aufgebaute Fenster, asynchrones HTTP mit Wiederholung.
Binärprotokolle	Gebaut, Testumgebung grün	R Protocol-Client von Hand: Demultiplexen über Template-IDs, Anmelde-Handshake, mehrteilige Antworten, 16 mitgelieferte Schemata, AES-256-GCM in zwei Laufzeiten mit identischem Format.
Windows-Distribution	Gebaut, Pipeline vorhanden	Tray-App mit lokalem Listener und Kontextisolation, Selbst-Update, globaler Hotkey; Inno Setup mit Terminal-Erkennung und Eintrag in fremde Konfigurationsdateien; Azure Trusted Signing mit Neuberechnung des Manifests.
Browser-Erweiterungen	Produktion	Vier Erweiterungen nach Manifest V3: Uploads im Service Worker, Side Panel, IndexedDB, eigener REST-Client, Firefox-Portierung, DOM-Automation.

Absicherung und Steuerung

Bereich	Stand	Beleg im Projekt
Testen	Nachgewiesen	314 Unit-Fälle, 89 End-zu-Ende-Fälle auf drei Bildschirmgrössen, Vertragstests gegen die API, Tests gegen die eigene Konfiguration.
Sicherheit	Nachgewiesen	Transport-Header, Schutz gegen Open Redirect, Entzug überflüssiger Ausführungsrechte, verschlüsselte Zugangsdaten, zeitkonstante Vergleiche, Trennung von Client- und Servergeheimnissen, eigene Audits mit Fundstellen.
Produkt und Strategie	Nachgewiesen, sieben Monate	Personas, Architekturentscheidungen, Roadmaps mit Erfolgskriterien, Nutzerinterviews mit Transkription, Wettbewerbs-Scorecards, Preismodelle mit Margenrechnung, Partnerverhandlungen bis zum unterzeichneten Vertrag.

Arbeitsweise, wie sie sich im Repository zeigt

- **Wie committet wird:** konventionelle Präfixe mit Bereichsangabe, 58 Prozent Korrekturen zu 33 Prozent neuen Funktionen. Gezählt auf dem Integrationszweig `feature/desktop-quick-log`, der die vollständige Historie trägt, ohne Merge- und Generator-Commits.
- **Schreibt auf, warum:** Kommentare im Code verweisen auf den konkreten Fehler, der zu einer Lösung geführt hat; die Entscheidungen halten Kontext, Alternativen und Folgen fest.
- **Audit statt Hoffnung:** der eigene Bezahlpfad zweimal mit Datei und Zeile durchgesehen, fünf Betriebsvorfälle mit Ursache dokumentiert, die eigene Warnschwelle an 59 Datensätzen kalibriert.
- **Wählt das Werkzeug nach dem Problem:** Vanilla JavaScript für Manifest V3, eine Node-Function für WebSocket und Protobuf, GitHub Actions gegen eine IP-Sperre, Inno Setup für die Terminal-Erkennung.
- **Kennt die eigenen Schulden:** Typstriktheit, zu grosse Komponenten, gemischte Datenzugriffstile, fehlende Test-Pipeline. Alles selbst benannt, nicht auf Nachfrage.
- **Korrigiert sich an Daten:** Die NinjaTrader-Strategie wurde nach eigener Recherche zurückgestuft, obwohl der Vertrag bereits unterzeichnet war, und eine abgelehnte Bewerbung wurde als Entscheidung mit Ursachenanalyse verarbeitet statt abgehakt.

Markt und externe Bestätigung

Die eigene Recherche schätzt 15 bis 30 Millionen aktive spekulative Trader weltweit, von denen 5 bis 10 Prozent ein Journal-Werkzeug nutzen; der kaufkraftgewichtete erreichbare Markt liegt zwischen 207 und 994 Millionen Dollar im Jahr, während die gesamte Kategorie heute geschätzt 5 bis 20 Millionen Dollar Jahresumsatz erwirtschaftet. Der grösste Wettbewerber liegt über 3 Millionen Dollar und wächst über die Reichweite seines Gründers, ein vergleichbares Solo-Projekt erreichte in drei Jahren rund 120'000 Dollar. Entrio ist nach dieser Recherche das einzige Journal, das Screenshot-Erfassung, strukturierte Psychologie pro Trade und KI-Auswertung verbindet; drei bis vier Anbieter weltweit versuchen den ersten Teil, keiner den zweiten.

Partner	Stand
NinjaTrader	Ecosystem Vendor Program: angenommen im Mai 2026, Vertrag unterzeichnet am 16. Juni 2026. Enterprise-Lizenz, Affiliate-Programm, Verzeichniseintrag, ein Newsletter-Platz mit 2.8 Millionen Empfängern, Webinare.
Rithmic	Developer Kit erhalten am 20. März 2026, Testumgebung Ende zu Ende grün am 6. Mai 2026. Produktion wartet auf das Konformitätsverfahren. Keine Gebühren für Anbieter.
Spotware	cTrader Open API lesend freigegeben am 16. Juni 2026.
FTMO	Antwort im März 2026, bestätigt die Abdeckungsstrategie. Die Nutzung der DXtrade-API ist von dieser Seite blockiert.
Trading-Schulen	Pilot mit fünf Schülern seit April 2026, davon drei im August aktiv. Eine zweite Schule mit 200 Schülern hat mündlich zugesagt. Über 40 weitere qualifizierte Kontakte recherchiert.
Programme	AWS Activate Credits im Februar 2026. Zwei Bewerbungen bei Frühphasen-Programmen wurden nicht angenommen; die Rückmeldung führte zur Neustrukturierung des Pitch-Decks.

Stand	7. September 2026 · live seit Februar 2026, vor dem kommerziellen Beweis
Kontakt	Gabriel Suter · kontakt@gabrielsuter.dev · gabrielsuter.dev
Produkt	app.entrio.cloud · www.entrio.cloud · gabriel@entrio.cloud
Quellcode	Private Repositories, Einsicht auf Anfrage
Grundlage	Vollständige Durchsicht aller fünf Repositories, ihrer Branches und der Knowledge Base am 5. September 2026; alle Zahlen stammen aus dem Code oder aus datierten Dokumenten.