



TRADING
JOURNAL

| TradingJournal

Vorstellung

Tipp zum Lesen am Bildschirm: Links mit der **rechten Maustaste** anklicken und „Link in neuem Tab öffnen“ wählen — ein normaler Klick öffnet die Seite im selben Fenster, und das PDF ist weg.

Version v1.4.0 · Stand 30.10.2026



TradingJournal

Das TradingJournal möchte ich euch zur Verfügung stellen.

Was ist das TradingJournal?

Das TradingJournal ist eine **Webanwendung, die auf dem eigenen Rechner läuft**. Bedient wird sie im Browser (Chrome, Edge, Firefox ...), so wie eine Internetseite — sie liegt aber nicht im Internet, sondern auf dem eigenen PC (oder einem eigenen Server). Es ist also weder eine App aus einem App-Store noch eine Online-Plattform, bei der man sich anmeldet und die Daten hochlädt.

Sie besteht aus zwei Teilen, die zusammenarbeiten:

- **TradingJournal-Weaver** — ein Programm ohne eigene Oberfläche, das im Hintergrund die Handelsdaten von Interactive Brokers (IBKR) abholt, aufbereitet und in einer Datenbank auf dem eigenen Rechner speichert.
- **TradingJournal** — die Weboberfläche, die diese Daten im Browser anzeigt: Dashboard, Kampagnen, Spreads, Auswertungen, Trade Log (das Entscheidungsjournal) und Psychologie-Modul.

Ich nutze es als Journal für meinen Options- und Aktienhandel bei Interactive Brokers: Es importiert meine IBKR-Trades automatisch, fasst sie zu durchgehenden Kampagnen und Spreads zusammen und macht Bestand, P&L und Kennzahlen auf einen Blick sichtbar. Der Schwerpunkt liegt auf allem, was generische Journal-Tools nicht abbilden: Options-Rollen, Spread-Strukturen, Zuteilungen aus Optionen in Aktienpositionen — ergänzt um ein Psychologie-Modul, das Ergebnis und Entscheidungsqualität bewusst trennt. Besonders ist, dass alles komplett lokal läuft, ohne Cloud-Zwang und ohne Abo. Die Idee dafür ist über ein Jahr alt, die aktuelle Version ist aus meinen eigenen Bedürfnissen gewachsen. Jetzt wage ich den nächsten Schritt und mache den Quellcode öffentlich (PolyForm Noncommercial).

Wichtig

Läuft nur unter Windows. Die gesamte Installation ist auf Windows zugeschnitten: die Schritt-für-Schritt-Anleitungen, die Hilfsskripte (PowerShell) und die Ablageordner für die IBKR-Berichte (standardmäßig `C:\Trade_XML\`, `C:\Trade_CSV\`). Andere Betriebssysteme unterstütze ich derzeit nicht.



Ausprobieren ohne Installation

Wer sich das TradingJournal erst einmal ansehen möchte, kann die öffentliche Demo nutzen — ganz ohne eigene Installation. Die Trades darin sind echt: Sie stammen aus zwei Echtgeld- und zwei Paper-Trading-Konten und wurden anonymisiert (Kontonummern ersetzt). Kurse, Stückzahlen und Ergebnisse sind unverändert. Nur die Einträge im Trade Log und im Psychologie-Modul sind beispielhaft ergänzt.

- **Adresse:** <https://204-168-201-48.sslip.io/login>
- **E-Mail:** demo@example.com
- **Passwort:** [Demo-Trading2026!](#)

Änderungen in der Demo werden nicht gespeichert. Die Demo ist ein freiwilliges Angebot und kann zeitweise nicht erreichbar sein.

Die Eckpunkte

- **Komplett lokal, volle Datenhoheit.** Ich nutze keinen Cloud-Dienst und keinen Drittanbieter-Server. Die App läuft auf meiner eigenen Maschine (lokaler PC oder ein eigener Server, falls vorhanden), meine Kontodaten bleiben dort.
- **Quellcode offen, Nutzung kostenlos — aber nicht kommerziell.** Ich habe beide Repositories öffentlich gemacht: [TradingJournal](#) (die Weboberfläche) und [TradingJournal-Weaver](#) (Import und Datenverarbeitung). Jeder darf das Journal für die eigenen Konten nutzen (auch Echtgeld), den Code lesen, anpassen und kostenlos weitergeben.
Nicht erlaubt ist, es zu verkaufen, als bezahlten Dienst anzubieten oder in ein kommerzielles Produkt einzubauen (PolyForm-Noncommercial-Lizenz, Details im README).
- **Zugeschnitten auf Interactive Brokers.** Ich importiere über IBKR Flex Query — andere Broker unterstütze ich aktuell nicht.
- **Kein Echtzeit-Tool.** Mein Import läuft mit ein bis zwei Tagen Nachlauf zum echten Handelstag. Für mich ist das Journal deshalb eher Dokumentation und Analyse im Rückblick als ein aktives Trading-Tool für den laufenden Tag.
- **Dokumentiert für den Einstieg.** Ich habe Setup-Anleitungen für Windows/MySQL/Laravel/Python geschrieben, dazu ein eingebautes Handbuch direkt in der App, das jeden Menüpunkt erklärt.
- **Hobbyprojekt, best effort.** Ich biete keinen Servicevertrag (z.B. wie schnell ich bei einer Fehlermeldung erste Schritte einleite) und auch kein Support-Team — nur mich, in meiner Freizeit. Das steht auch im README.
- **KI-unterstützt entwickelt.** Das Projekt wurde mit Claude von Anthropic umgesetzt.
- **5 getrennte Environments (Umgebungen).** MONEY, PAPER, TEST, AUTOM_MONEY und AUTOM_PAPER halten Echtgeld, Paper-Trading, Testdaten und automatisierten Handel sauber auseinander.
- **USD und EUR.** Trades erscheinen in ihrer Handelswährung (bei US-Werten also USD). Im Hintergrund führt der Import alle Beträge zusätzlich in EUR mit — umgerechnet mit dem



Wechselkurs, den IBKR zu jedem Trade mitliefert. Beide Werte gleichzeitig sehe ich derzeit nur an wenigen Stellen, vor allem auf der Detailseite eines Trades. Noch offen: eine Umschaltung zwischen USD und EUR per Klick bzw. die EUR-Werte überall mit anzuzeigen. Reine Währungsumtausch-Trades (USD↔EUR-Wechsel) haben eine eigene Übersicht, inklusive Zusammenfassung, wenn IBKR eine Order in mehreren Teilausführungen gefüllt hat.

- **Konfigurierbares Dashboard.** Über ein Auswahlfenster blende ich Kacheln ein und aus, wie ich sie gerade brauche.

Wie mein Workflow aussieht

- 1 **Import der Daten von IBKR** — eine Flex Query liefert mir die Rohdaten meines Kontos als XML (durch Zustellung per Mail und/oder Abfrage per API)
- 2 **Import & Verarbeitung (Python)** — ich parse die Rohdaten (XML-Dateien), führe sie zu Kampagnen und Spreads zusammen und berechne laufenden Bestand und P&L.
- 3 **Ansehen & Verstehen (Weboberfläche)** — im Dashboard, bei Kampagnen, Spreads und in den Symbol-/Zeitraum-Auswertungen sehe ich, wie sich mein Portfolio entwickelt.
- 4 **Reflektieren** — im Trade Log halte ich fest, warum ich einen Trade eingegangen bin, im Psychologie-Modul, ob ich mich an meine eigenen Regeln gehalten habe.

Den Import kann ich jederzeit von Hand anstoßen (ein einziger Befehl) oder automatisch laufen lassen, etwa täglich über die Windows-Aufgabenplanung. Weil IBKR die Berichte erst nach Handelsschluss erstellt, sehe ich meine Trades mit dem erwähnten Nachlauf von ein bis zwei Tagen. Die App ist für mich deshalb Auswertung und Journal-Pflege — kein Live-Cockpit für den Handelstag selbst.

Pro & Contra

Dafür spricht:

- Ich habe volle Kontrolle über meine eigenen Finanzdaten, nichts verlässt meine eigene Maschine
- Ich muss nichts mehr händisch nachtragen — die IBKR-Daten fließen automatisiert ein
- Ich bilde Options-Spezifika ab, die generische Journal-Tools nicht können: Kampagnen mit Rollen, Spread-Strukturen, Zuteilungen aus Optionen in Aktienpositionen
- Ich trenne Ergebnis und Entscheidungsqualität durch das Psychologie-Modul
- Kostenlos und quelloffen, keine Abo-Falle



- Den Code habe ich nach bestem Wissen und Gewissen erstellt und geprüft

Dagegen spricht:

- Ich unterstütze nur Interactive Brokers als Datenquelle — für andere Broker ist es aktuell unbrauchbar
- Kein Echtzeit-Tool: Ich sehe meine Positionen mit ein bis zwei Tagen Nachlauf, nicht live.
- Mein Setup ist kein Einzeiler: Windows, MySQL, PHP/Laravel, Python und mehrere Konfigurationsdateien müssen zusammenspielen. Wer sich vor der Kommandozeile scheut, wird hier nicht glücklich
- Ich habe nur begrenzte Testabdeckung — solide Kernlogik, aber keine durchgängige, produktionsreife QA
- Ich biete keinen Support und keine Roadmap-Garantie — ich baue weiter, wenn ich Zeit und Lust habe
- Reine Weboberfläche für meinen Schreibtisch, keine mobile App

Persönliche Entwicklung

- Angefangen habe ich vor über einem Jahr. Die Dokumentation meiner Trades durch die Eingabe in Excel-Sheets war mir immer zu mühsam. Dokumentation auf Papier noch mehr. Dann bin ich auf die Idee gekommen mir die Daten von IBKR zu holen (IBKR hat meine Trades ohnehin als strukturierte Datensätze - Flex Query). Damit habe ich meine ersten Versuche mit Python, einer MySQL-Datenbank und Google Sheets als Darstellungstool gestartet. Das hat eine Weile gut funktioniert — bis ich angefangen habe, verstärkt mit Optionen zu handeln. Da ich vor allem Spreads handle, wegen der Margin-Vorteile, wurde die Darstellung/Dokumentation in einer Tabelle immer umständlicher. Außerdem wollte ich eine verfolgbare Dokumentation der Wheel-Strategie (Bonus-Strategie), was mit Googlesheets nicht möglich war. Über ein anderes Projekt bin ich auf Laravel gestoßen und habe beschlossen, mir damit eine echte Oberfläche zu bauen, statt weiter gegen Tabellenkalkulationen zu kämpfen — so ist die nächste Version des TradingJournal entstanden.
- Die Kampagnen, also die durchgehende Geschichte aller Trades zu einem Symbol statt einer losen Liste von Einzeltrades, waren dabei ursprünglich als reine Dokumentation für mein Wheel-Trading (Bonus-Strategie) gedacht und sind von dort aus gewachsen.
- Weiters war es mir immer wichtig: kein Geld für Tools und/oder Abos ausgeben. Die verschiedenen Tools (im Internet) sind sehr mächtig und können oft viel mehr als mein Tradingjournal, doch alles läuft Online und meine Daten stehen IRGENDWO ... Kontoauszüge sind Finanzdaten — die wollte ich ohnehin nicht auf einen fremden Host hochladen, sondern in meiner eigenen, lokalen Umgebung behalten.
- Dazu kam ein weiterer Punkt, der mir beim eigenen Handeln immer wieder auffiel: Das Ergebnis eines Trades sagt mir nichts darüber, ob die Entscheidung dahinter gut war. Ein Gewinn aus Disziplin und ein Gewinn aus einem Impuls sehen in meinem P&L identisch aus. Also habe ich neben der reinen Buchhaltung ein Modul gebaut, das genau das für mich trennt.



Das Buch von Annie Duke – Thinking in Bets – hat mich dazu inspiriert: Entscheidungsqualität ≠ Ergebnisqualität. Ein schlechter Trade kann durch Glück Geld verdienen — eine gute Entscheidung kann durch Pech Geld verlieren. Langfristig zählt nur der Prozess.

- Und ein technischer Punkt: Ich programmiere gerne und bringe das Grundwissen mit, aber erst über den KI-Boom und Vibe-Coding bin ich so tief in Laravel eingestiegen (ich habe das Projekt mit Claude von Anthropic umgesetzt). Ohne das wäre ich heute weit davon entfernt, eine funktionierende Oberfläche zu haben.

Status

Das TradingJournal ist im täglichen Einsatz und wird laufend weiterentwickelt. Mit der Veröffentlichung des Quellcodes wünsche ich mir Rückmeldungen aus der Praxis — gerade auch zu Lücken in der Dokumentation.

Wohin mit Fragen und Fehlermeldungen? Am besten ins Forum, in dem ich das TradingJournal vorstelle — dort beantworte ich alles direkt. Fragen, die öfter kommen, sammle ich und fasse sie zu einer FAQ (häufig gestellte Fragen) zusammen. Wer ein GitHub-Konto hat, kann Fehler, Ideen und Beispieldaten auch als Issue melden (es gibt Vorlagen dafür) oder Änderungen als Pull Request vorschlagen.

Kleines Begriffs-Lexikon

Begriff	Bedeutung
API	Programmierschnittstelle — hier: der Weg, über den das Programm die Berichte direkt bei IBKR abfragt, statt auf eine E-Mail zu warten.
Best effort	„Nach bestem Bemühen“ — ohne Zusage, ob und wie schnell etwas erledigt wird.
Cloud	Rechner und Speicher bei einem fremden Anbieter im Internet. Das Gegenteil von „lokal“.
Anonymisiert	Persönliche Merkmale (hier: die Kontonummern) sind entfernt oder ersetzt, sodass die Daten keiner Person mehr zugeordnet werden können.
Dashboard	Startseite mit den wichtigsten Kennzahlen auf einen Blick, aufgeteilt in Kacheln (kleine Kästchen mit je einer Zahl).
Environment (Umgebung)	Ein getrennter Datenbereich, z. B. für Echtgeld oder Paper-Trading — die Daten vermischen sich nicht. In der App oben rechts als farbiges Badge zu sehen (MONEY, PAPER ...).
FAQ	<i>Frequently Asked Questions</i> — Sammlung häufig gestellter Fragen mit Antworten.



Flex Query	Bei IBKR selbst zusammengestellter Bericht, der alle Trades in einer maschinenlesbaren Form liefert.
Generisch	Allgemein gehalten, für alles und jeden gedacht — und dadurch nicht auf Besonderheiten (hier: Optionen) zugeschnitten.
Issue	Eintrag auf GitHub, mit dem man einen Fehler meldet, eine Frage stellt oder eine Idee vorschlägt.
Kampagne	Die zusammenhängende Geschichte aller Trades zu einem Symbol — vom ersten Einstieg bis zum vollständigen Ausstieg, inklusive Rollen und Zuteilungen.
Kommandozeile	Fenster, in dem man Befehle eintippt statt zu klicken (unter Windows z. B. PowerShell).
Laravel, Python, MySQL	Laravel ist das Baukastensystem für die Weboberfläche (in der Programmiersprache PHP), Python die Programmiersprache für den Import, MySQL die Datenbank, in der alles gespeichert wird.
Lizenz	Regeln, was man mit dem Programm und seinem Quellcode tun darf. Hier: PolyForm Noncommercial — private Nutzung ja, kommerzielle Nutzung nein.
Lokal / selbstgehostet	Läuft auf dem eigenen Rechner bzw. Server, nicht bei einem fremden Anbieter.
Modal	Ein Fenster, das sich über der Seite öffnet (z. B. zum Ein- und Ausblenden der Dashboard-Kacheln) und wieder geschlossen wird, bevor man weiterarbeitet.
P&L	<i>Profit and Loss</i> — Gewinn und Verlust.
Paper-Trading	Handel mit Spielgeld auf einem Übungskonto, ohne echtes Risiko.
Pull Request	Änderungsvorschlag auf GitHub: Jemand schlägt konkrete Code-Änderungen vor, die ich prüfen und übernehmen kann.
Quellcode	Der lesbare Programmtext, aus dem das Programm besteht. „Quelloffen“ heißt: jeder kann ihn einsehen.
README	Die Beschreibungsdatei, die auf der Startseite eines Repositorys angezeigt wird.
Repository (Repo)	Ein Projektordner auf GitHub, in dem der gesamte Quellcode samt Änderungsgeschichte liegt. Das TradingJournal besteht aus zwei Repos.
Rollen	Eine laufende Optionsposition schließen und gleichzeitig eine neue mit späterem Verfall und/oder anderem Basispreis eröffnen.
Spread	Kombination mehrerer Optionen auf denselben Basiswert, die zusammen gekauft bzw. verkauft werden (z. B. Bull Put Spread, Iron Condor).
Trade Log	Das Entscheidungsjournal der App (Menüpunkt <i>Log</i>): Vor oder nach einem Trade festhalten, warum man ihn eingeht und wie er geplant ist.
Vibe-Coding	Programmieren im Dialog mit einer KI: Man beschreibt, was entstehen soll, die KI schreibt den Code, man prüft und steuert nach.



Wheel-Strategie	Optionsstrategie: Puts verkaufen, bei Zuteilung die Aktien übernehmen und darauf Calls verkaufen, bis die Aktien wieder abgegeben werden.
XML	Ein Textformat für strukturierte Daten — so liefert IBKR die Flex-Query-Berichte.
Zuteilung (Assignment)	Der Verkäufer einer Option muss liefern bzw. abnehmen — aus der Option wird eine Aktienposition.