



TRADING
JOURNAL

| TradingJournal

Benutzerhandbuch

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



Inhalt

Überblick — vom Trade bis zur Auswertung 7

- | | |
|---|----|
| 1. Trading bei IBKR | 9 |
| 2. IBKR liefert die Daten: Flex Query und Kontoauszug | 9 |
| 3. Einlesen: Abholen und Import | 10 |
| 4. Verarbeiten: FIFO-Matching und Kampagnen | 11 |
| 5. Sichern: die Sammeldatei (Summary) | 14 |
| 6. Welche Daten entstehen | 15 |
| 7. Darstellung im TradingJournal | 17 |

Rundgang durch die App 19

- | | |
|---|----|
| 1. Kopfzeile — was auf jeder Seite gilt | 19 |
| 2. Dashboard | 20 |
| 3. Aktien | 22 |
| 4. Optionen | 23 |
| 5. Kampagnen | 24 |
| 6. Spreads | 26 |
| 7. Log — das Entscheidungsjournal | 29 |
| 8. Psychologie | 31 |
| 9. Analyse | 32 |



IBKR einrichten — Flex Queries und Kontoauszug

34

1. Wo die Einstellungen liegen	34
2. Flex Query für die Trades anlegen	35
3. Query-ID finden	41
4. Tägliche Zustellung per E-Mail	42
5. Abruf per Web-API (optional)	43
6. Kontoauszug (CSV) für Kontostand und Rendite	45
7. Zusammenfassung: was wohin gehört	49

Windows vorbereiten

50

Inhalt	50
1. Was du am Ende hast	50
2. Ein paar Begriffe vorab	51
3. Git installieren	51
4. XAMPP installieren (Datenbank + Webserver)	52
5. Composer installieren	54
6. Node.js installieren	54
7. Python installieren	55
8. Projektordner anlegen und Code herunterladen	55
9. Alles überprüfen	57
10. Wie geht's weiter?	57



TradingJournal-Weaver installieren 58

Inhalt	58
1. Projekt herunterladen	58
2. Python-Arbeitsumgebung einrichten	59
3. Konfigurationsdatei anlegen	59
4. Datenbank-Tabellen anlegen	62
5. Automatisierung einrichten	62
6. Ersten Testlauf machen	63
7. Praktisch: Schnellzugriff einrichten	64
8. Alte Daten nachholen (einmalig)	64
Zusammenfassung — künftig starten	69
Weiterführende Dokumentation	69

TradingJournal installieren 70

Inhalt	70
1. Projekt herunterladen	70
2. Konfigurationsdatei anlegen	71
3. Datenbank-Tabellen anlegen	71
4. Ersten Benutzer anlegen	72
5. Optik der Seite bauen	73
6. Starten und öffnen	73
7. Praktisch: Schnellzugriff einrichten	73
Zusammenfassung — künftig starten	74
Weiterführende Dokumentation	74



Erster Start — UI-Erstkonfiguration 75

- | | |
|--|----|
| 1. Anmelden | 75 |
| 2. Der Environment-Switcher (oben in der Navigation) | 75 |
| 3. Mehrere Personen nutzen dasselbe TradingJournal | 76 |
| 4. Dashboard-Kacheln anpassen | 77 |

Alltag mit dem TradingJournal 78

- | | |
|---|----|
| 1. Daten holen und verarbeiten | 78 |
| 2. Nach dem Import: ein typischer Durchgang | 85 |
| 3. Kampagnen: offen, wartend, geschlossen | 86 |
| 4. Wenn etwas nicht stimmt | 87 |
| 5. Mehrere Benutzer | 89 |

Eigener Server 90

- | | |
|--|----|
| 1. Was gleich bleibt | 90 |
| 2. Was anders ist | 91 |
| 3. Wie die Daten auf den Server kommen | 91 |
| 4. Automatisch laufen lassen (Windows-Aufgabenplanung) | 92 |
| 5. Updates | 93 |



Updates einspielen

94

Was ein Update macht — und was nicht

94

Update starten

94

Was das Skript im Einzelnen tut

95

Nach dem Update: neue Einstellungen?

96

Sicherheitsupdates für PHP

96

Welche Version läuft gerade?

96

Wenn etwas schiefgeht

97

Fehlerbehebung und häufige Fragen

98

1. Zuerst nachsehen: die Protokolle

98

2. Installation

98

3. Daten abholen (Mail und Web-API)

99

4. Import

100

5. Anzeige im Journal

101

Überblick — vom Trade bis zur Auswertung

Dieses Kapitel beschreibt den Weg eines Trades: von der Order bei Interactive Brokers (IBKR) bis zur Zahl im TradingJournal. Wer den Ablauf einmal verstanden hat, kann später auch einschätzen, *warum* ein Trade noch nicht sichtbar ist, warum eine Position offen bleibt oder woher eine Kennzahl stammt.

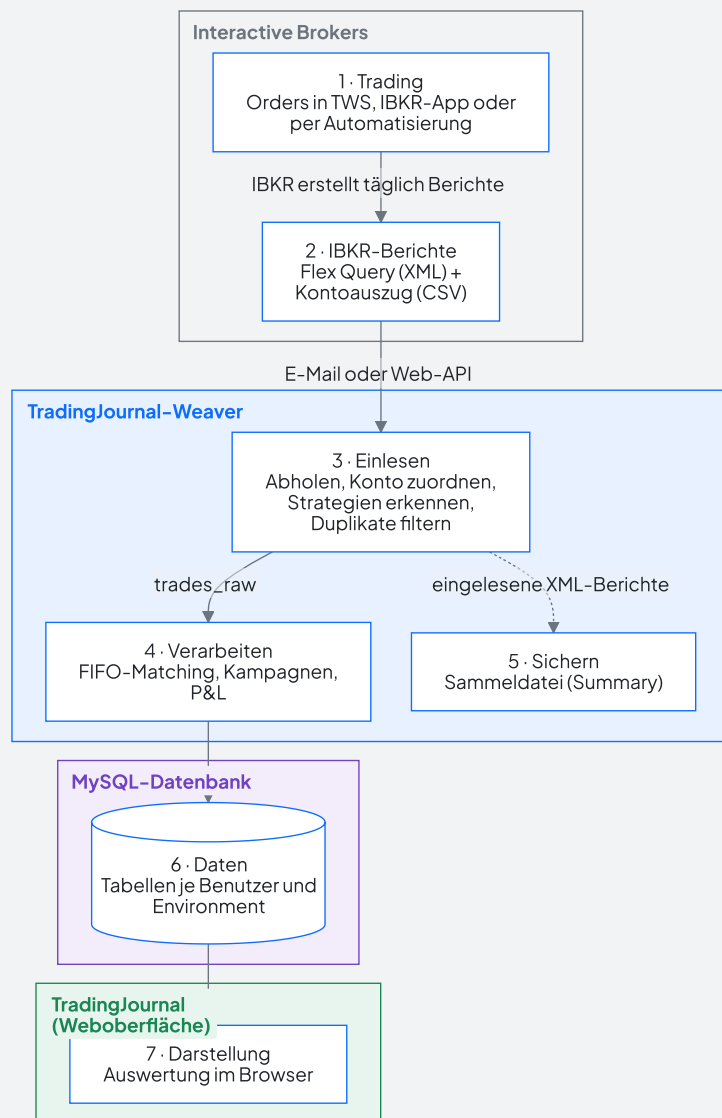
Das TradingJournal ist eine **Webanwendung, die auf dem eigenen Rechner läuft** (oder auf einem eigenen Server). Bedient wird sie im Browser wie eine Internetseite — sie liegt aber nicht im Internet. Es ist also weder eine App aus einem App-Store noch eine Online-Plattform, auf die man seine Daten hochlädt.

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 werden derzeit nicht unterstützt.

Beteiligt sind zwei Programme, die sich eine MySQL-Datenbank teilen:

- **TradingJournal-Weaver** (Python) holt die Berichte von IBKR, liest sie ein und rechnet sie aus. Es *schreibt* die Trading-Daten.
- **TradingJournal** (Weboberfläche) zeigt diese Daten an. Es *liest* die Trading-Daten und schreibt nur seine eigenen Einträge (Trade Log, Psychologie, Benutzer).



Die Farben zeigen, wer zuständig ist: **Interactive Brokers** (grau) erzeugt die Daten, **TradingJournal-Weaver** (blau, Python) holt, liest und verarbeitet sie, die **MySQL-Datenbank** (violett) speichert sie, und das **TradingJournal** (grün, Weboberfläche) stellt sie dar. Die Nummern entsprechen den Abschnitten dieses Kapitels.

Hinweis

Das Journal ist **kein Echtzeit-Tool**. IBKR erstellt die Berichte erst nach Handelsschluss; zwischen Trade und Anzeige liegen deshalb typischerweise ein bis zwei Tage.

1. Trading bei IBKR

Gehandelt wird wie gewohnt — in der Trader Workstation (TWS), in der IBKR-App oder über ein automatisiertes Handelssystem. Für das Journal muss **nichts** am Handel geändert werden.

- **Nur lesender Zugriff:** Das Journal verbindet sich nie mit der Handelsschnittstelle und kann keine Orders auslösen. Es verarbeitet ausschließlich Berichte, die IBKR ohnehin erstellt.
- **Mehrere Konten:** Jedes IBKR-Konto (z. B. Echtgeld, Paper, ein separates Konto für automatisierten Handel) wird später genau einem **Environment** zugeordnet. So bleiben Echtgeld, Übungs- und Testdaten strikt getrennt:

Environment	Typische Verwendung
MONEY	Echtgeld-Konto, manueller Handel
PAPER	Paper-Trading-Konto (Simulation)
AUTOM_MONEY	Echtgeld-Konto für automatisierten Handel
AUTOM_PAPER	Paper-Konto für automatisierten Handel
TEST	Ausprobieren und Testen, ohne echte Daten zu berühren

2. IBKR liefert die Daten: Flex Query und Kontoauszug

IBKR bietet mit **Flex Queries** frei konfigurierbare Berichte an, die automatisch erstellt und per E-Mail zugestellt oder über eine Web-Schnittstelle abgerufen werden können. Das Journal nutzt drei Arten von Berichten:

Bericht	Format	Zustellung	Inhalt	Wofür
Flex Query	XML	täglich per E-Mail	Trades, Verfall/Ausübung/Zuteilung von Optionen, Kapitalmaßnahmen (Splits)	Hauptquelle für alle Trades
Flex Query per Web-API	XML	auf Abruf (Token + Query-ID)	Trades, Verfall/Ausübung/Zuteilung von Optionen, Kapitalmaßnahmen (Splits)	Zweiter, unabhängiger Kanal — falls eine Mail ausbleibt
Kontoauszug (Activity Statement)	CSV	täglich per E-Mail	Kontostand, Cash, Tagesperformance (TWRR)	Kontoverlauf und Rendite auf dem Dashboard

**Achtung**

Der Kontoauszug wird nur per E-Mail zugestellt. Es gibt keine Möglichkeit, ihn per Abruf (Web-API) anzufordern. Rückwirkend lässt er sich nur Tag für Tag von Hand im Kundenportal herunterladen.

Für **jedes IBKR-Konto**, dessen Daten verarbeitet werden sollen, wird eine eigene Flex Query eingerichtet, dazu optional ein eigener Kontoauszug — ohne ihn fehlen nur die Kontostands-Kacheln auf dem Dashboard. Welche Einstellungen dafür nötig sind, zeigt Schritt für Schritt das Kapitel IBKR einrichten. Wie eine Flex Query technisch aufgebaut ist und welche Felder der Import daraus liest, beschreibt das technische Handbuch (Aufbau einer Flex Query).

3. Einlesen: Abholen und Import

Das Einlesen erledigt ein einziger Befehl in `TradingJournal-Weaver`, der alle Schritte nacheinander ausführt und typischerweise einmal täglich automatisch läuft (Windows-Aufgabenplanung):

```
python run_pipeline.py
```

Schritt 1 — Abholen

- `fetch_emails` meldet sich per IMAP am eingetragenen Postfach an, sucht ausschließlich Mails von IBKR. Die Dateianhänge (XML) werden am Rechner lokal im Verzeichnis `C:\Trade_XML\incoming_mail\`, die Kontoauszüge (CSV) im Verzeichnis `C:\Trade_CSV\incoming\` gespeichert.
- `fetch_api` ruft — falls konfiguriert — dieselben Trades zusätzlich über die IBKR-Web-API ab und speichert sie im Verzeichnis `C:\Trade_XML\incoming_api\`.
- Dateien können auch von Hand in diese Ordner kopiert werden; der weitere Ablauf ist derselbe.

Schritt 2 — Import

Der Import liest jede XML-Datei und überträgt die Trades unverändert in die Datenbank (Tabellenname: `trades_raw`). Dort werden sie zu diesem Zeitpunkt als „noch nicht verarbeitet“ markiert. Während des Einlesens passiert Folgendes:

- 1 **Konto zuordnen** — Anhand der letzten vier Ziffern der IBKR-Kontonummer (steht in den Daten) werden die Daten einem Benutzer und Environment zugeordnet. Unbekannte Konten werden mit einer Warnung übersprungen, nie „irgendwo“ einsortiert.



- 2 **Teilausführungen bündeln** — Wurde eine Order in mehreren Teilen ausgeführt, werden die Fills derselben Order (`ibOrderID`) zusammengefasst.
- 3 **Options-Strategien erkennen** — In der TWS ist ein Spread ein eigenes, synthetisches Finanzinstrument (Combo-Order). In den Flex Queries liefert IBKR jedoch jede Option als eigenen Trade. Das System setzt das synthetische Instrument heuristisch wieder zusammen: Beine mit gleicher Order-ID oder nahezu gleichzeitig ausgeführte Beine (dieselben 3 Sekunden) werden gruppiert und als Strategie benannt, etwa Bull Put / Bear Put / Bear Call / Bull Call Spread, Iron Condor, Straddle, Strangle, Calendar Spread, Diagonal Spread oder Butterfly.
- 4 **Options-Ereignisse zuordnen** — Verfall, Ausübung, Zuteilung und Barausgleich (z. B. bei SPX) werden der passenden automatischen IBKR-Buchung zugeordnet.
- 5 **Währungstausch abtrennen** — Reine Devisen-Trades (z. B. EUR.USD) landen zusätzlich in der eigenen Datenbanktabelle `trades_fx`.
- 6 **Duplikate filtern** — Jeder Trade erhält eine Prüfsumme. Ist die Prüfsumme in der Datenbank noch nicht vorhanden, wird der Trade gespeichert. Kommt derselbe Trade erneut an (über Mail und Web-API, oder weil eine Datei doppelt eingelesen wird), wird er übersprungen.

Die Kontoauszüge (CSV) werden verarbeitet und es wird ein Snapshot (aktueller Wert des Depots) angelegt.

Tipp

`trades_raw` enthält die Trades genau so, wie IBKR sie geliefert hat. Bei Unklarheiten ist die Seite **Raw Data** (unter *Analyse*) im Journal deshalb der erste Anlaufpunkt: Steht ein Trade dort, ist er angekommen — fehlt er, liegt das Problem vor dem Import (Bericht, Mail, Kontozuordnung).

4. Verarbeiten: FIFO-Matching und Kampagnen

Der nächste Schritt arbeitet alle noch nicht verarbeiteten Trades aus der Datenbanktabelle `trades_raw` in zeitlicher Reihenfolge ab und entscheidet für jeden Trade, was er bedeutet. Dabei passieren zwei Dinge gleichzeitig: Das **FIFO-Matching** ordnet jeden Trade den offenen Positionen zu



und berechnet Gewinn und Verlust (4.1), und jede Veränderung wird als Buchung in der **Kampagne** des Symbols festgehalten (4.2).

Jeder Trade wird in einer eigenen Datenbank-Transaktion verarbeitet: Entweder sind alle zugehörigen Einträge geschrieben oder keiner — auch bei einem Absturz entstehen keine halben Zustände. Alle Geldbeträge werden als exakte Dezimalzahlen gerechnet, nicht als Gleitkommazahlen, damit sich keine Rundungsfehler aufsummieren.

4.1 FIFO-Matching

Wie in den IBKR-Steuerberichten wird nach *First In, First Out* gerechnet: Ein Verkauf schließt immer zuerst die älteste offene Tranche. Dadurch stimmen Gewinn und Verlust mit den offiziellen IBKR-Auswertungen überein. Eine **Tranche** ist dabei ein einzelner Kauf (bzw. bei Short-Positionen ein einzelner Verkauf) mit eigenem Einstandspreis und Datum.

Beispiel mit vier Trades in AAPL (Gewinne brutto, ohne Kommission):

Datum	Trade	Was passiert	Offen danach
03.03	Kauf 100 zu 150 \$	neue Tranche A	A: 100 zu 150 \$
10.03	Kauf 50 zu 160 \$	neue Tranche B, Tranche A bleibt unverändert	A: 100, B: 50 zu 160 \$
20.03	Verkauf 120 zu 170 \$	schließt zuerst A komplett (100 Stück, +2.000 \$), dann 20 Stück von B (+200 \$)	B: 30 zu 160 \$
25.03	Verkauf 80 zu 165 \$	schließt den Rest von B (30 Stück, +150 \$), die übrigen 50 Stück eröffnen eine Short-Position	Short 50 zu 165 \$

Jeder geschlossene Teil wird ein eigener **abgeschlossener Trade** mit eigenem Einstandspreis, eigener Haltedauer und eigenem P&L. Der Verkauf vom 20.03. erscheint unter *Geschlossene Positionen* deshalb als zwei Zeilen (100 und 20 Stück). Offene Tranchen stehen in der Datenbanktabelle `trades_open`, abgeschlossene Trades in `trades_close`.

Daneben gibt es einige Sonderfälle:

- **Aktiensplit** — Menge und Einstandspreis aller offenen Tranchen werden angepasst, es entsteht kein abgeschlossener Trade.
- **Verfall, Ausübung, Zuteilung** — Die Option wird als abgeschlossener Trade mit der passenden Abschlussart gespeichert (*Expired, Assigned, CashSettled*). Bei einer Zuteilung wird die Aktienbuchung wie ein normaler Trade verarbeitet — beim zugeteilten Put kommen die Aktien als neue Tranche hinzu, beim zugeteilten Covered Call gehen sie aus dem Bestand.
- **Devisentausch** (z. B. EUR.USD) — kein FIFO, eigene Übersicht.



Hinweis

Kein Forex-Trading. Der Devisentausch wird nur als Umtausch festgehalten — etwa, um Euro in US-Dollar für den Aktienhandel zu tauschen. Devisenhandel als eigene Strategie (Währungspositionen eröffnen und schließen, Gewinn/Verlust aus Kursbewegungen) wertet das Journal **nicht** aus: Es gibt dafür weder Positionen noch Gewinn/Verlust oder Kampagnen.

Wie die Tabellen dabei im Einzelnen geschrieben werden, beschreibt das technische Handbuch (Architektur und Import-Pipeline, Abschnitt „FIFO-Matching: Die Fälle“).

Für jeden abgeschlossenen Trade werden **P&L**, **ROI** (bezogen auf das eingesetzte Kapital) und **Haltedauer** berechnet. Die Beträge (P&L, Cashflow, Kommission) stehen in der Handelswährung und zusätzlich umgerechnet in Euro — mit dem Wechselkurs, den IBKR zu jedem Trade mitliefert.

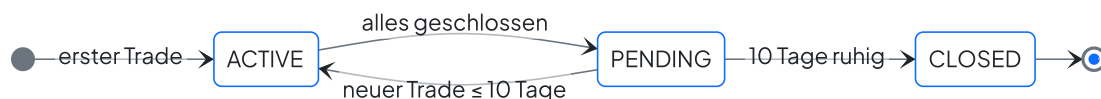
Wichtig

Alle gespeicherten P&L-Werte sind **brutto**, also ohne Kommissionen. Die Kommissionen werden separat gespeichert; das Journal zeigt deshalb überall Brutto- und Netto-P&L (Netto = Brutto + Kommission, da IBKR Kommissionen negativ ausweist).

4.2 Kampagnen und Ledger

FIFO beantwortet die Frage „Was hat dieser einzelne Trade gebracht?“. Viele Strategien bestehen aber aus einer Kette von Trades, die nur zusammen Sinn ergeben. Deshalb werden alle Trades eines Symbols, die zeitlich zusammenhängen, zu einer **Kampagne** gebündelt — Aktien und Optionen auf dasselbe Underlying gehören dabei zur selben Kampagne. So erscheint z. B. eine Wheel-Strategie (Put verkauft → Zuteilung → Covered Call → neuer Put) als **eine** zusammenhängende Position statt als viele Einzeltrades.

Lebenslauf einer Kampagne — Der erste Trade in einem Symbol eröffnet eine Kampagne (*ACTIVE*). Ist nichts mehr offen, wartet sie 10 Tage (*PENDING*): Kommt in dieser Zeit ein neuer Trade, läuft dieselbe Kampagne weiter; sonst wird sie abgeschlossen (*CLOSED*), und der nächste Trade in dem Symbol beginnt eine neue. Die 10 Tage sind der Standard; wer eine andere Frist möchte, stellt sie in der `.env` von `TradingJournal-Weaver` ein (`MAX_GAP_DAYS`).



Das Ledger: die Buchhaltung jeder Kampagne — Für jede Kampagne wird ein *Ledger* geführt, ein Kassenbuch im Sinne der Buchhaltung. Jede Veränderung erzeugt darin eine eigene Buchung: Eröffnen, (teilweises) Schließen, Verfall, Ausübung oder Zuteilung einer Option, Barausgleich,



Aktiensplit. Eine FIFO-Teilung wie am 20.03. im Beispiel oben ergibt zwei Buchungen, eine je Tranche. Jede Buchung enthält:

- die **Art** der Buchung (z. B. `OPT_PUT_SELL` = Put verkauft, `OPTION_EXPIRATION` = Option verfallen) und die **Mengenänderung**,
- den **Cashflow** (Geldbewegung auf dem Konto), die **Kommission** und — bei Schließungen — den **realisierten P&L**,
- **laufende Summen**: aufgelaufener P&L, Kommissionen und Erlöse (Proceeds) der Kampagne bis zu dieser Buchung,
- Währung, Wechselkurs und die Beträge in Euro,
- einen Verweis auf den Ursprungs-Trade. Zusammengehörige Buchungen, etwa die zugeteilte Option und die dadurch gelieferten Aktien, sind miteinander verknüpft.

Beispiel einer Wheel-Kampagne (vereinfacht, ohne Kommissionen):

Buchung	Art	Menge	Cashflow	P&L	P&L kumuliert
Put XYZ 50 verkauft	<code>OPT_PUT_SELL</code>	-1 Option	+150 \$	—	0 \$
Put zugeteilt	<code>OPTION_EXERCISE</code>	+1 Option	—	+150 \$	150 \$
100 Aktien zu 50 \$ eingebucht	<code>STK_BUY</code>	+100 Stk.	-5.000 \$	—	150 \$
Call XYZ 55 verkauft	<code>OPT_CALL_SELL</code>	-1 Option	+80 \$	—	150 \$
Call verfallen	<code>OPTION_EXPIRATION</code>	+1 Option	—	+80 \$	230 \$

Buchungen werden im laufenden Betrieb nur angehängt, nie nachträglich verändert. Aus dem Ledger entstehen im Journal der **P&L-Verlauf**, die **Tages-Übersicht** und die **Transaktions-Historie** jeder Kampagne, außerdem die Auswertungen unter *Stats* und *Aktivität*. Auch der Bestand einer Kampagne (Spalte *Bestand* in der Transaktions-Historie) ergibt sich aus der Summe aller Mengenänderungen.

5. Sichern: die Sammeldatei (Summary)

Wann? Als letzter Schritt jedes Pipeline-Laufs, also direkt nach der Verarbeitung und damit normalerweise einmal täglich — für jeden Benutzer und jedes Environment außer `TEST`.

Was passiert? Nach dem Import liegen die eingelesenen XML-Berichte (aus Mail und Web-API) im Ordner `C:\Trade_XML\<environment>\<benutzer>\processed\`. Der Schritt `summary` nimmt die dort neu hinzugekommenen Berichte und hängt ihre Tagesberichte an eine **Sammeldatei** an:



```
C:\Trade_XML\<environment>\summary_<präfix><environment>.xml
```

Jeder Berichtstag wird nur einmal aufgenommen, leere Berichte (Tage ohne Trades) werden übersprungen, und die Datei bleibt nach Datum sortiert. So wächst über die Zeit eine vollständige Historie aller Trades in einer einzigen Datei.

Wozu? Muss die Datenbank einmal neu aufgebaut werden — etwa nach einem Fehler oder wenn eine neue Version anders rechnet —, werden aus der Sammeldatei alle Trades erneut eingelesen und verarbeitet, ohne dass die ursprünglichen Mails noch vorhanden sein müssen. Wie das geht, beschreibt das technische Handbuch (Befehlsreferenz).

Wichtig

Die Sammeldatei enthält nur die **Trades** aus den Flex Queries, keine Kontoauszüge (CSV). Sie ist die wichtigste Datei für eine Datensicherung: Wer sie aufbewahrt, kann seine Trade-Historie jederzeit wiederherstellen.

6. Welche Daten entstehen

Die Daten sind auf zwei Ebenen getrennt: **Jedes Environment hat eine eigene Datenbank**, und **jeder Benutzer hat darin seine eigenen Tabellen**, erkennbar am Präfix vor dem Tabellennamen. Ein Beispiel mit zwei Benutzern: Anna handelt mit Echtgeld und auf einem Paper-Konto, Tom nur auf Paper.



```
MySQL-Server
|
|— ibkr_laravel          Weboberfläche: Benutzerkonten (gemeinsam für alle)
|   |— users            Anna → Präfix anna_, Tom → Präfix tom_
|
|— ibkr_money           Environment MONEY
|   |— anna_trades_raw
|   |— anna_trades_close
|   |— anna_campaign_ledger
|   |— ... alle weiteren Tabellen mit anna_
|
|— ibkr_paper           Environment PAPER
|   |— anna_trades_raw
|   |— anna_...
|   |— tom_trades_raw
|   |— tom_...
|
|— ibkr_test            Environment TEST
|— ibkr_autom_money     Environment AUTOM_MONEY
|— ibkr_autom_paper     Environment AUTOM_PAPER
```

- Tabellen entstehen nur dort, wo sie eingerichtet werden: Tom hat in `ibkr_money` keine Tabellen, weil er kein Echtgeld-Konto verbindet (eingerichtet im Kapitel TradingJournal-Weaver installieren, Schritt „Datenbank-Tabellen anlegen“).
- Das Präfix muss an zwei Stellen gleich sein: in `users.toml` von TradingJournal-Weaver (dorthin schreibt der Import) und beim Benutzerkonto der Weboberfläche (von dort liest das Journal).
- Beim Anmelden bestimmt das Benutzerkonto das Präfix, der **Environment-Schalter** wählt die Datenbank. Anna sieht in *PAPER* also `ibkr_paper` mit den Tabellen `anna_...`, nicht die von Tom. Einzige Ausnahme: Ein Administrator kann bewusst auf einen anderen Benutzer wechseln — aber nur, wenn die Weboberfläche im lokalen Modus läuft (`APP_ENV=local`).

Je Benutzer und Environment gibt es folgende Tabellen:

Tabelle	Inhalt	Entsteht in
<code>trades_raw</code>	Trades genau so, wie IBKR sie geliefert hat	Import
<code>trades_fx</code>	Devisentausch-Trades	Import
<code>symbols</code>	Stammdaten der Instrumente (Symbol, ISIN, Kontrakt Daten)	Import
<code>account_snapshots</code>	Tageswerte des Kontos: Nettovermögen, Cash, Tagesrendite	Import (Kontoauszug-CSV)
<code>trades_open</code>	Offene Positionen als einzelne FIFO-Tranchen	Verarbeitung
<code>trades_close</code>	Abgeschlossene Trades mit P&L, ROI, Haltedauer und Abschlussart	Verarbeitung
<code>trading_campaigns</code>	Kampagnen je Symbol mit Status ACTIVE / PENDING / CLOSED	Verarbeitung
<code>campaign_ledger</code>	Jede Buchung einer Kampagne mit laufendem P&L	Verarbeitung

Die Tabellen oben schreibt ausschließlich TradingJournal-Weaver; die Weboberfläche liest sie nur. Die folgenden Tabellen gehören genauso jedem Benutzer und Environment einzeln (z. B. `anna_trade_planner` in `ibkr_paper`). Der Unterschied: Ihr Inhalt stammt nicht von IBKR, sondern aus den eigenen Eingaben — es sind die **einzigen Tabellen, die du selbst in der Weboberfläche anlegst, änderst und löschst**:

Tabelle	Inhalt
<code>trade_planner</code> , <code>trade_planner_legs</code>	Trade Log: geplante und reflektierte Entscheidungen
<code>psychology_*</code>	Psychologie-Modul: Regelwerke, Journal-Einträge, Regel-Auswertung

7. Darstellung im TradingJournal

Das TradingJournal liest die Tabellen des angemeldeten Benutzers im gewählten Environment. Der **Environment-Schalter** oben in der Navigation wählt also die Datenbank; der Benutzer-Login bestimmt, welche Tabellen (Präfix) gelesen werden. Neue Daten sind sichtbar, sobald die Pipeline gelaufen ist — ein Neuladen der Seite genügt.



Bereich	Zeigt	Datenquelle
Dashboard	Kennzahlen (P&L, Profit Factor, Max Drawdown, Rendite), Kontoverlauf	<code>trades_close</code> , <code>trades_open</code> , <code>account_snapshots</code>
Aktien / Optionen → Offene Positionen	aktueller Bestand je Tranche	<code>trades_open</code>
Aktien / Optionen → Geschlossene Positionen	abgeschlossene Trades mit P&L, ROI, Haltedauer	<code>trades_close</code>
Spreads (Vertical Spreads, Iron Condors)	erkannte Options-Strategien mit allen Beinen	<code>trades_open</code> , <code>trades_close</code>
Kampagnen	Kampagnen-Liste und Detailseite mit P&L-Verlauf und Ledger	<code>trading_campaigns</code> , <code>campaign_ledger</code>
Analyse → Symbole	alle Kampagnen eines Symbols im Überblick	<code>trading_campaigns</code> , <code>campaign_ledger</code>
Analyse → Stats / Aktivität	Auswertungen je Zeitraum, chronologische Buchungsliste	<code>campaign_ledger</code>
Analyse → FX-Trades	Währungstausch-Trades	<code>trades_fx</code>
Analyse → Raw Data	Trades im Originalzustand von IBKR	<code>trades_raw</code>
Log	eigene Planung und Reflexion, verknüpfbar mit Kampagnen	<code>trade_planner</code> + Kampagnen
Psychologie	Regelwerke und Regel-Einhaltung je Trade	<code>psychology_*</code> + Trades

Währungen: Angezeigt wird die Handelswährung (bei US-Werten USD). Die Euro-Werte führt TradingJournal-Weaver im Hintergrund mit; beide Währungen gleichzeitig zeigen derzeit nur die **Detailseiten geschlossener Trades**. Die Kontostand-Kacheln im Dashboard zeigen Euro, so wie IBKR sie im Kontoauszug liefert; das Bargeld steht getrennt in einer USD- und einer EUR-Kachel. Eine Umschaltung zwischen USD und EUR per Klick ist noch nicht umgesetzt.

Tipp

Die Detailseiten einzelner Trades verlinken auf den zugehörigen Eintrag in **Raw Data**. So lässt sich ein abgeschlossener Trade bis zu den Originaldaten aus dem IBKR-Bericht zurückverfolgen.

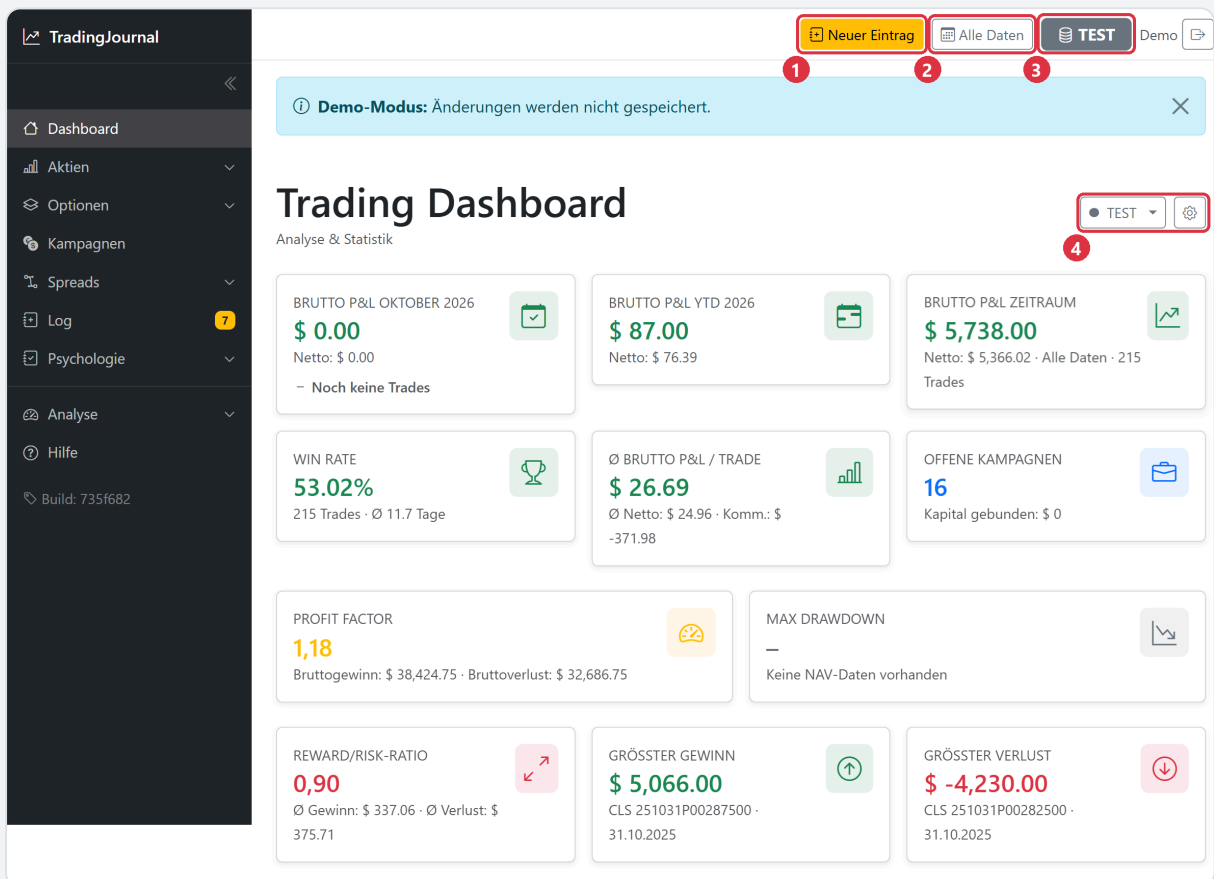
Rundgang durch die App

Dieses Kapitel führt einmal durch alle Bereiche des TradingJournal: wofür sie gedacht sind und welche Kennzahlen, Tabellen und Diagramme dich dort erwarten. Die Reihenfolge folgt dem Menü links. Kurze Erklärungen zu jeder Seite gibt es auch direkt in der App unter **Hilfe**.

Hinweis

Die Bilder stammen aus der Demo-Version. Sie zeigt echte, anonymisierte Trades aus zwei Echtgeld- und zwei Paper-Trading-Konten. Bei dir sehen Zahlen, Symbole und Umfang natürlich anders aus.

1. Kopfzeile — was auf jeder Seite gilt



The screenshot shows the Trading Journal Dashboard in Demo mode. The interface includes a left sidebar menu, a top header with navigation buttons, and a main dashboard area with various performance metrics. Numbered callouts highlight specific features:

- 1:** 'Neuer Eintrag' button in the top header.
- 2:** 'Alle Daten' button in the top header.
- 3:** 'TEST' button in the top header.
- 4:** 'TEST' dropdown menu in the top right corner.

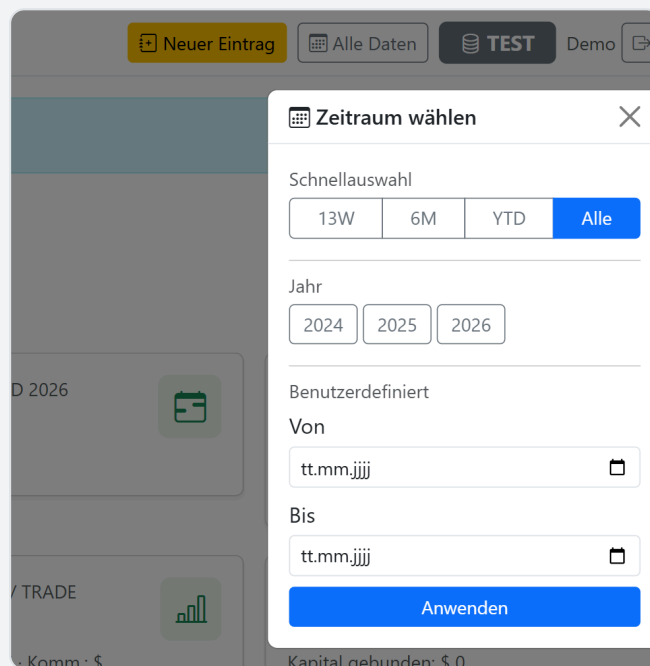
Trading Dashboard
Analyse & Statistik

Metric	Value	Details
BRUTTO P&L OKTOBER 2026	\$ 0.00	Netto: \$ 0.00, Noch keine Trades
BRUTTO P&L YTD 2026	\$ 87.00	Netto: \$ 76.39
BRUTTO P&L ZEITRAUM	\$ 5,738.00	Netto: \$ 5,366.02 · Alle Daten · 215 Trades
WIN RATE	53.02%	215 Trades · Ø 11.7 Tage
Ø BRUTTO P&L / TRADE	\$ 26.69	Ø Netto: \$ 24.96 · Komm.: \$ -371.98
OFFENE KAMPAGNEN	16	Kapital gebunden: \$ 0
PROFIT FACTOR	1,18	Bruttogewinn: \$ 38,424.75 · Bruttoverlust: \$ 32,686.75
MAX DRAWDOWN	—	Keine NAV-Daten vorhanden
REWARD/RISK-RATIO	0,90	Ø Gewinn: \$ 337.06 · Ø Verlust: \$ 375.71
GRÖSSTER GEWINN	\$ 5,066.00	CLS 251031P00287500 · 31.10.2025
GRÖSSTER VERLUST	\$ -4,230.00	CLS 251031P00282500 · 31.10.2025

Dashboard mit Kopfzeile



Nr.	Element	Wofür
1	Neuer Eintrag (gelb)	öffnet sofort ein neues Formular im Log, um eine Trade-Idee festzuhalten, solange sie frisch ist
2	Zeitraum (hier „Alle Daten“)	grenzt fast jede Liste, Kennzahl und jedes Diagramm auf ein Zeitfenster ein; die Beschriftung zeigt die aktuelle Auswahl
3	Environment-Badge	zeigt, in welchem Datenbereich du gerade bist (MONEY , PAPER , TEST , AUTOM_MONEY , AUTOM_PAPER)
4	Environment-Umschalter und Zahnrad (auf dem Dashboard)	wechselt den Datenbereich bzw. wählt aus, welche Kacheln das Dashboard zeigt — siehe Erster Start — UI-Erstkonfiguration



Zeitraum wählen

Der **Zeitraum** bietet eine Schnellauswahl (letzte 13 Wochen, 6 Monate, seit Jahresbeginn oder alle Daten), ganze Jahre oder einen frei gewählten Zeitraum. Die Auswahl bleibt bestehen, bis du sie änderst.

Das Menü links lässt sich über die Doppelpfeile oben einklappen; die Zahl neben **Log** zählt Kampagnen, die noch nicht im Log dokumentiert sind (siehe Abschnitt 7).

2. Dashboard

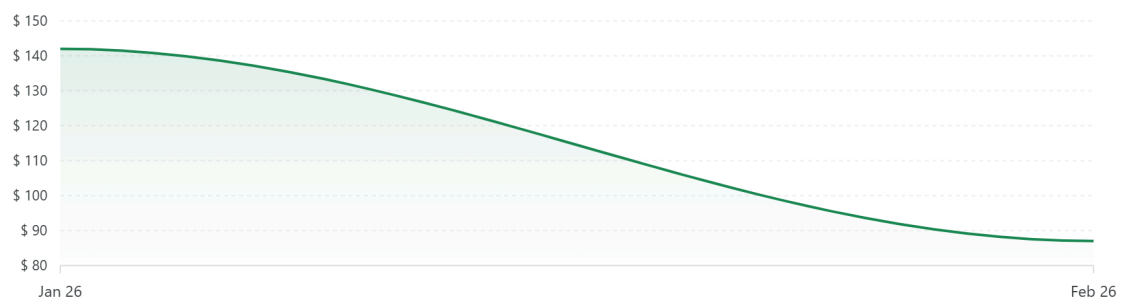
Die erste Seite nach dem Start: Hier siehst du auf einen Blick, wie es um das Portfolio steht.



- **Kennzahlen-Kacheln** (oberes Bild): P&L für den laufenden Monat, das Jahr und den gewählten Zeitraum, Trefferquote (Win Rate), Ø P&L pro Trade, offene Kampagnen, Profit Factor, Max Drawdown, Reward/Risk, größter Gewinn und Verlust sowie — wenn Kontoauszüge importiert werden — Cash und Kontostand mit Rendite. Welche Kacheln erscheinen, stellst du über das Zahnrad ein.
- **Diagramme:** kumulierter P&L-Verlauf, P&L nach Strategie und nach Option/Aktie, monatlicher P&L der letzten 12 Monate und der Kontostand-Verlauf.
- **Optionen, die diese Woche verfallen**, und **aktuelle offene Positionen** ganz unten. Die Positionstabelle zeigt alle offenen Positionen (Anzahl im Kopf, bei vielen Einträgen scrollbar) und verlinkt auf *Offene Aktien* und *Offene Optionen*.

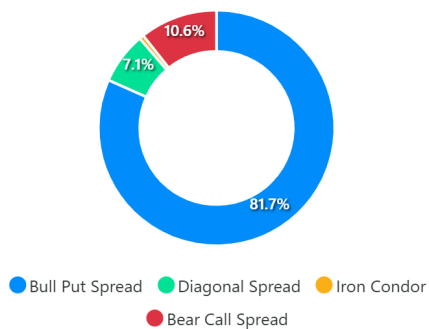
Kumulativer P&L 2026

monatlich



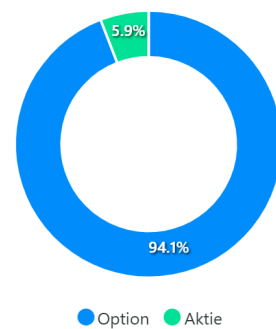
P&L nach Spreads

Alle Daten



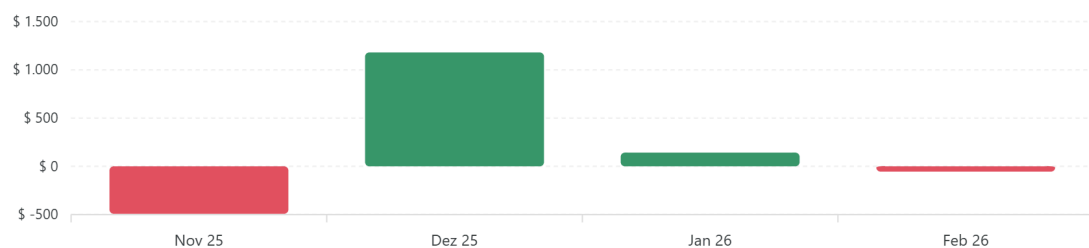
P&L nach Option/Aktie

Alle Daten



Monatlicher P&L

letzte 12 Monate



Dashboard: Diagramme



Hinweis

P&L-Werte stehen überall **brutto** (ohne Kommissionen) mit dem **Netto**-Wert (nach Kommissionen) darunter.

3. Aktien

Der Aktienbestand, im Menü getrennt nach **Offene Positionen** und **Geschlossene Positionen**.

Offene STK-Positionen

Offene Positionen

ANZAHL STK POSITIONEN

15

#

GEBUNDENES KAPITAL

17842,70 \$



SUMME KOMMISSIONEN

-15,47 \$



Symbol

Symbol, Beschreibung...

Reset

offene STK-Positionen

15

Symbol	Eröffnet ↓	Stücke	Einstiegspreis	Gesamtwert	NetCash	Buy/Sell	
BMWd BAYERISCHE MOTOREN WERKE AG	27.10.2025 vor 11 Monaten	1,00	81,68	81,68	-84,68	BUY	
CAH CARDINAL HEALTH INC	20.10.2025 vor 11 Monaten	-6,00	157,55	-945,30	944,80	SELL	
ON ON SEMICONDUCTOR	17.10.2025 vor 1 Jahr	-100,00	52,00	-5200,00	5199,98	SELL	
COR CENCORA INC	01.10.2025 vor 1 Jahr	-3,00	312,86	-938,58	937,58	SELL	
CVS CVS HEALTH CORP	01.10.2025 vor 1 Jahr	-15,00	75,81	-1137,15	1136,15	SELL	

Offene Aktienpositionen

- **Offene Positionen:** Kacheln für Anzahl, gebundenes Kapital und Kommissionen; die Tabelle zeigt Symbol, Eröffnungsdatum, Stückzahl, Einstiegspreis, Gesamtwert, Cashflow (NetCash) und Kauf/Verkauf. Über das Suchfeld filterst du nach Symbol oder Beschreibung.
- **Geschlossene Positionen:** zusätzlich Haltedauer, Brutto- und Netto-P&L, ROI und Strategie.



Detailseite (Augen-Symbol rechts): Preis, Menge, Cashflow, gehaltene Tage, Trade- und Zeitangaben. Stammt eine Position aus einer **Options-Zuteilung**, zeigt ein eigener Block, welche Prämie dafür schon kassiert wurde und welcher effektive Einstandspreis sich daraus ergibt. Bei geschlossenen Positionen steht das Ergebnis in USD und umgerechnet in EUR.

4. Optionen

Aufgebaut wie die Aktien-Ansicht, ergänzt um alles, was eine Option ausmacht.

✓ Geschlossene OPT-Positionen

geschlossene Positionen

Alle historischen Daten

ANZAHL OPTIONS POSITIONEN

178

#

TOTAL PREMIUM

5402,00 \$



GESAMTE KOMMISSIONEN

-298,98 \$



Suche

Symbol, Underlying...

Put/Call

Alle

Strategie

Alle

Pro Seite

25

Reset

geschlossene OPT-Positionen

178

Underlying	Buy/Sell	Strike	Contracts	Eröffnet →	Daysheld	Preis	Status	Brutto	Kommission	Strategie
				Geschlossen ↓				P&L / Netto		
SPX SPXW 260210P06970000	BUY	P 6,970.00	1	10.02.2026 → 10.02.2026 geschlossen vor 8 Monaten	0	7,15	CashSettled	2.104,00 \$ netto: 2.102,36 \$	-1,64 \$ -	
SPX SPXW 260210P06975000	SELL	P 6,975.00	1	10.02.2026 → 10.02.2026 geschlossen vor 8 Monaten	0	8,60	CashSettled	-2.459,00 \$ netto: -2.460,64 \$	-1,64 \$ -	
SPX SPXW 260204P06845000	BUY	P 6,845.00	1	04.02.2026 → 04.02.2026 geschlossen vor 8 Monaten	0	6,90	Expired	-690,00 \$ netto: -691,64 \$	-1,64 \$ -	
SPX SPXW	BUY	C 6,905.00	1	04.02.2026 → 04.02.2026 geschlossen	0	6,30	Expired	-630,00 \$ netto: -631,64 \$	-1,64 \$ -	

Geschlossene Optionspositionen

- Strike mit **P/C** (Put/Call) und Verfallsdatum; bei offenen Optionen ein deutlicher Hinweis, sobald eine Option bereits verfallen ist, die kassierte Prämie und eine Spalte „**50%-Ziel**“ als Richtwert, ab wann sich ein vorzeitiges Schließen lohnen kann.
- Bei geschlossenen Optionen die **Abschlussart** in der Spalte *Status*: regulär geschlossen, *Expired* (verfallen), *Assigned* (zuteilt) oder *CashSettled* (Barausgleich).



- Filter nach Put/Call und Strategie.

Die **Detailseite** zeigt zusätzlich Multiplikator, Strategie und die Wechselkurse bei Eröffnung und Schließung.

5. Kampagnen

Eine Kampagne bündelt alles, was zusammenhängend zu einem Symbol gehandelt wurde — zum Beispiel ein Wheel aus verkauften Puts, Aktienzuteilung und Covered Calls. Statt vieler loser Einzeltrades erzählt sie die ganze Geschichte einer Position (wie Kampagnen entstehen und enden, beschreibt Überblick — vom Trade bis zur Auswertung).

✓ Kampagnen

Kampagnen

Alle historischen Daten

Suche über alle Kampagnen hinweg

Symbol, Underlying...

Reset

Offen (Aktiv & Pending)

Geschlossen

Alle

17 Kampagnen

Brutto PnL: 713,30 \$

Netto PnL: 660,12 \$

BMW 27.Oct.2025 Log Aktiv

Stock: BMW (2025-10-27)

Brutto PnL	Bestand
-	1
Netto: -3,00 \$	
Proceeds	Commission
-81,68 \$	-3,00 \$

BWEN 19.Aug.2025 Log Aktiv

Stock: BWEN (2025-08-19)

Brutto PnL	Bestand
-	-357
Netto: -1,85 \$	
Proceeds	Commission
760,41 \$	-1,85 \$

BZUN 15.Sep.2025 Log Aktiv

Stock: BZUN (2025-09-15)

Brutto PnL	Bestand
-	209
Netto: -1,05 \$	
Proceeds	Commission
-1.009,47 \$	-1,05 \$

CAH 08.Sep.2025 Log Aktiv

Stock: CAH (2025-09-08)

Brutto PnL	Bestand
45,30 \$	-6
Netto: 43,30 \$	
Proceeds	Commission
990,60 \$	-2,00 \$

COMM 19.Aug.2025 Log Aktiv

Stock: COMM (2025-08-19)

Brutto PnL	Bestand
-	64
Netto: -1,00 \$	
Proceeds	Commission
-1.001,60 \$	-1,00 \$

COR 01.Oct.2025 Log Aktiv

Stock: COR (2025-10-01)

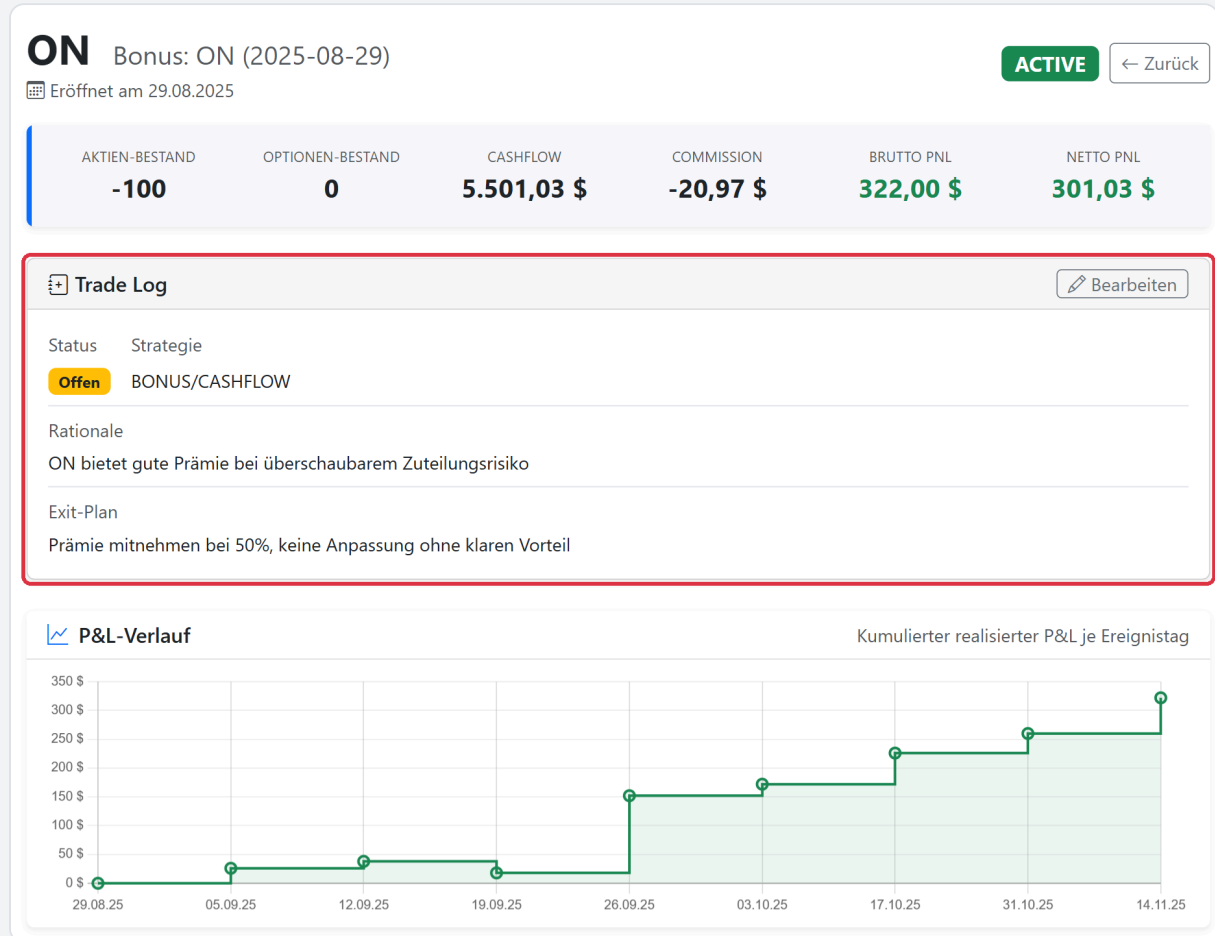
Brutto PnL	Bestand
-	-3
Netto: -1,00 \$	
Proceeds	Commission
938,58 \$	-1,00 \$

Kampagnen-Übersicht



Liste: eine Karte je Kampagne mit Symbol, Startdatum, Status (*Aktiv, Pending, Geschlossen*), Brutto- und Netto-P&L, Bestand, Erlösen (Proceeds) und Kommission. Das **Log-Badge** zeigt, ob die Kampagne schon im Log dokumentiert ist (grün hinterlegt = ja). Oben umschalten zwischen offenen, geschlossenen und allen Kampagnen; das Suchfeld durchsucht alle. Daneben filtert eine zweite Knopfreihe nach **Art**: *Aktien* (nur Aktien), *Wheel* (Aktien und Optionen, mit oder ohne Zuteilung), *Optionen* (nur Einzeloptionen) und *Spreads* (mindestens ein Leg gehört zu einem Spread). Die Art wird aus allen Buchungen der Kampagne bestimmt und steht als Etikett auf jeder Karte.

Detailseite (Klick auf eine Karte):



Kampagne: Kennzahlen, Trade Log und P&L-Verlauf

- Kennzahlen: Aktien- und Optionsbestand, Cashflow, Kommission, Brutto- und Netto-P&L.
- **Eröffnungen & Kommentare** (markiert): je Eröffnung ein Block — alle zum selben Zeitpunkt eröffneten Legs (Put, Spread, Iron Condor) oder ein Aktienkauf. Die Zahlen kommen aus den Buchungen: Legs, Prämie in Dollar, Prämie beim Schließen, Gebühren, Ausgang (geschlossen, verfallen, zugeteilt) und P&L. Darunter steht dein Kommentar oder der Knopf **Kommentar**; im Kopf die Angabe, wie viele Eröffnungen kommentiert sind. Eine Zuteilung ist keine eigene Eröffnung, und Aktien-P&L steht nur in der Summe der Kampagne, weil eine Aufteilung auf Put und Call willkürlich wäre. Ein Wheel erscheint so als Folge seiner Puts und Calls.
- **P&L-Verlauf** als Stufendiagramm: der aufgelaufene realisierte P&L je Ereignistag.
- **Tages-Übersicht**: je Tag Buchungen, Cashflow, Kommission, P&L und Ereignisse wie Verfall.



- **Transaktions-Historie:** jede Buchung im Klartext (z. B. „Put Prämie: 5x – ON 44,5 P ...“), mit Mengenänderung, P&L und laufendem Bestand, gruppiert nach Tag mit Tagessumme. Zusammengehörige Buchungen (etwa beide Beine einer Rollposition) sind farbig hinterlegt.

Transaktions-Historie								
41 Buchungen								
ZEIT	EREIGNIS	BESCHREIBUNG	Δ MENGE	CASHFLOW	COMMISSION	BRUTTO PNL	NETTO PNL	BESTAND
Freitag, 14. November 2025								
16:20	OPTION EXPIRATION	ON 44,5 P 14.11.25 verfallen	+5	-	-	262,00 \$	262,00 \$	-100
16:20	OPTION EXPIRATION	ON 43,5 P 14.11.25 verfallen	-5	-	-	-200,00 \$	-200,00 \$	-100
Tagessumme				-	-	62,00 \$	62,00 \$	
Freitag, 7. November 2025								
14:10	OPT. PUT SELL	Put Prämie: 5x - ON 44,5 P 14.11.25 @ 0,52	-5	259,93 \$	-2,07 \$	—	-2,07 \$	-100
14:10	OPT. PUT BUY	Put Kauf: 5x - ON 43,5 P 14.11.25 @ 0,40	+5	-202,06 \$	-2,06 \$	—	-2,06 \$	-100
Tagessumme				57,87 \$	-4,13 \$	—	-4,13 \$	
Freitag, 31. Oktober 2025								
16:20	OPTION EXPIRATION	ON 50 P 31.10.25 verfallen	+1	-	-	52,00 \$	52,00 \$	-100
16:20	OPTION EXPIRATION	ON 49 P 31.10.25 verfallen	-1	-	-	-35,00 \$	-35,00 \$	-100
16:20	OPTION EXPIRATION	ON 50 P 31.10.25 verfallen	+1	-	-	52,00 \$	52,00 \$	-100
16:20	OPTION EXPIRATION	ON 49 P 31.10.25 verfallen	-1	-	-	-35,00 \$	-35,00 \$	-100
Tagessumme				-	-	34,00 \$	34,00 \$	

Kampagne: Transaktions-Historie

6. Spreads

Optionsstrategien aus mehreren Beinen — Vertical Spreads, Iron Condors und Verwandte — werden hier als **eine** Konstruktion betrachtet, nicht als Handvoll Einzelpositionen. Das Menü bietet vier Blickwinkel: offene und geschlossene Spreads, Vertical Spreads, Iron Condors.



Options Spreads

Alle historischen Daten

ANZAHL SPREADS

61

44 Gewinner / 16
VerliererBRUTTO GEWINN-
VERLUST

4529,36 \$

WIN RATE

72.1%

Ø BRUTTO / SPREAD

74,25 \$

Suche Underlying

SPY, AAPL, TSLA...

Strategie

Alle Strategien

Pro Seite

25

Reset

Geschlossene Spreads

61

Strategie	Underlying	Strikes	Expiry	Eröffnet	Geschlossen	Profit (SELL / BUY)	% of max	Netto	Ø Tage
Bull Put Spread	SPX	P: 6970/6975 5-wide	10.02.2026	10.02.2026	10.02.2026	-355,00 \$ -2459,00 / 2104,00	-250%	-358,28 \$	0
Bull Put Spread	SPX	P: 6845/6850 5-wide	04.02.2026	04.02.2026	04.02.2026	160,00 \$ 850,00 / -690,00	102%	156,72 \$	0
Bear Call Spread	SPX	C: 6900/6905 5-wide	04.02.2026	04.02.2026	04.02.2026	140,00 \$ 770,00 / -630,00	102%	136,72 \$	0
Diagonal Spread	TLT	C: 87 0-wide	16.01.2026	19.12.2025	16.01.2026	142,00 \$ 142,00 / 0,00	101%	141,23 \$	28
Bull Put Spread	IBKR	P: 50/60 10-wide	19.12.2025	28.10.2025	19.12.2025	76,00 \$ 91,00 / -15,00	103%	73,90 \$	52

Spreads-Übersicht

- **Kacheln:** Anzahl (mit Gewinnern/Verlierern), Brutto-Ergebnis, Trefferquote, Ø je Spread.
- **Tabelle:** Strategie, Underlying, Strikes mit Breite, Verfall, Eröffnung/Schließung, Gewinn (darunter Verkaufs- und Kaufseite einzeln), „% of max“ — wie viel vom maximal möglichen Gewinn erreicht wurde —, Netto und Ø Haltedauer.

Detailseite: Kennzahlen (Ergebnis, erhaltene/gezahlte Prämie, Haltedauer, Kommission), alle Beine in einer Tabelle, ein Performance-Block (ROI, Gewinn pro Tag) und die Verknüpfung zum Psychologie-Eintrag — fehlt er noch, lässt er sich hier direkt anlegen. Bei einem **Iron Condor** zeigt eine Strukturgrafik Put- und Call-Spread nebeneinander, je mit Strikes, Breite und — bei geschlossenen — Ergebnis.



🔗 Iron Condor auf SPX

[← Zurück](#)[✓ Geschlossen](#)GESAMTGEWINN/-
VERLUST**30,00 \$**

Netto: 46,32 \$



Premium

+5061,84 \$ erhalten**-5048,16 \$** gezahlt

HALTEDAUER

32 Tage20.10.2025 –
21.11.2025

KOMMISSIONEN

-16,32 \$

4 Legs



☰ Alle Legs (4)

Symbol	Type	Strike	Expiry	Action	Contracts	Preis Open	Preis Close	Profit	Tag	Kommission
SPX 251121P05870000	P	5,870.00	21.11.2025	BUY	3	11.70	0.00	-3.510,00 \$	32	-4,08 \$
SPX										
SPX 251121P05880000	P	5,880.00	21.11.2025	SELL	3	11.70	0.00	3.510,00 \$	32	-4,08 \$
SPX										
SPX 251121C07130000	C	7,130.00	21.11.2025	SELL	3	5.20	0.00	1.560,00 \$	32	-4,08 \$
SPX										
SPX 251121C07140000	C	7,140.00	21.11.2025	BUY	3	5.10	0.00	-1.530,00 \$	32	-4,08 \$
SPX										
TOTAL:								30,00 \$		-16,32 \$

📈 Performance

ROI 0.59%

Profit / Tag 0.94 \$

Max Risk 0.00 \$

Strategie Iron Condor

Underlying SPX

Legs 4

Strategy ID f733c42d-3d76-448a-adf7-71ef7d377de9

Spread-Detailseite

☰ Iron Condor Struktur

PUT Spread

BUY 5,870.00 11.70 \$

SELL 5,880.00 11.70 \$

Width: 10.00 | P/L: 0.00 \$

CALL Spread

SELL 7,130.00 5.20 \$

BUY 7,140.00 5.10 \$

Width: 10.00 | P/L: 30.00 \$

📈 Profit Range: 5,880.00 – 7,130.00 | Range Width: 1,250.00

Iron Condor: Strukturgrafik



7. Log — das Entscheidungsjournal

Der Kern eines Trading-Journals: Nicht nur das Ergebnis zählt, sondern der Gedanke dahinter. Hier hältst du fest, **warum** ein Trade eingegangen wurde — mit welchem Plan, welcher Erwartung und welchem Ausstiegsszenario.

Trade Log

+ Neuer Eintrag

ⓘ 7 Kampagne(n) ohne Log-Eintrag (letzte 30 Tage, inkl. offene)

SPX BONUS/CASHFLOW PENDING	Dokumentieren
04.02.2026 → 10.02.2026	
COR DIRECT_INVEST ACTIVE	Dokumentieren
01.10.2025 → 01.10.2025	
BZUN DIRECT_INVEST ACTIVE	Dokumentieren
15.09.2025 → 15.09.2025	
TBIL DIRECT_INVEST ACTIVE	Dokumentieren
08.09.2025 → 08.09.2025	
ZEPP DIRECT_INVEST ACTIVE	Dokumentieren
25.08.2025 → 25.08.2025	
VERB DIRECT_INVEST ACTIVE	Dokumentieren
19.08.2025 → 19.08.2025	
BWEN DIRECT_INVEST ACTIVE	Dokumentieren
19.08.2025 → 19.08.2025	

OFFENE EINTRÄGE
10

GESCHLOSSENE EINTRÄGE
48

REALISierter G/V
5024,70 \$

Underlying
SPY, AAPL...

Status
Alle

Pro Seite
25

≡ Log-Einträge

58

Trade Log mit Hinweis auf undokumentierte Kampagnen

- **Hinweisbox** (markiert): Kampagnen der letzten 30 Tage ohne Log-Eintrag. **Dokumentieren** öffnet den Kommentar zur ersten Eröffnung der Kampagne; weitere Eröffnungen kommentierst du auf der Kampagnen-Seite. Die Zahl neben Log im Menü zählt genau diese Kampagnen.
- **Kacheln** für offene und geschlossene Einträge sowie den realisierten Gewinn, darunter die Tabelle aller Einträge mit Filter nach Underlying und Status. Bei verknüpften Einträgen kommen Status und P&L aus den Buchungen.



Eintrag zu einer Eröffnung:



Neuer Eintrag

[← Zurück](#)

① Trade-Kopfdaten

Datum *	Underlying *	Kurs bei Open	Gebühren Open	IVP	IVR
18.10.2026 📅	SPY, AAPL...			0-100	0-100
Strategie		Exit-Plan / Regeln			
Iron Condor, BuPS...		TP bei 50%, SL bei 2x Prämie			
Rationale					
Warum dieser Trade? Welche Marktbedingung? Was erwartest du?					

🔗 Legs

[+ Leg hinzufügen](#)

Leg	DTE	Strike	Delta	Prämie	Gebühren	Max. Risiko
Noch keine Legs — klicke "Leg hinzufügen"						

Neuer Log-Eintrag

- **Oben die Zahlen aus den Buchungen**, nur zur Anzeige: Legs mit Strike, Prämie, Delta und DTE, Prämie der Eröffnung und beim Schließen in Dollar, Gebühren, maximales Risiko beim Vertical, Ausgang und P&L (**brutto**, wie in Kampagnen, Spreads und Statistik). Du trägst sie nicht ein, sie können also nicht von den Buchungen abweichen.
- **Dein Kommentar:** Strategie, Exit-Plan bzw. Regeln, Begründung (*Rationale*) und Notizen. Jede Eröffnung hat ihren eigenen Eintrag; ein Folgetrade im selben Underlying bekommt also einen eigenen Kommentar.
- **Kurs und IV:** Ist im Weaver der optionale Schritt **enrich** eingerichtet (siehe Alltag mit dem TradingJournal, Abschnitt 1, „Delta und IV-Rang automatisch“), stehen Kurs des Basiswerts und IVP/IVR bei Eröffnung und Schließung als berechnete Werte im Feld („aus IBKR berechnet“). Was du selbst einträgst, gilt; ohne **enrich** bleiben die Felder für deine Eingabe.

Eintrag ohne Kampagne: Vor dem Import, oder wenn keine Kampagne passt, steht das volle Formular bereit: Kopfdaten, Legs und über **Trade schließen** die Abschlusssdaten, alles von Hand (Gebühren positiv, Ergebnis brutto). Findet das Formular zum Underlying eine Kampagne, kannst du den Eintrag verknüpfen. Er gehört dann zur noch nicht kommentierten Eröffnung, die seinem Datum am nächsten liegt, und die Zahlen kommen ab dann aus den Buchungen.



8. Psychologie

Das Modul trennt bewusst das **Ergebnis** eines Trades von der Frage, **wie** er zustande kam. Ein Gewinn aus einer disziplinierten Entscheidung und einer aus einem Impuls sehen im P&L gleich aus — hier wird sichtbar, welcher es war. Das Menü hat vier Punkte:

- **Regelwerke:** die Maßstäbe — Regelwerke (Name, Strategie, aktiv/inaktiv) mit Einzelregeln und ihrem Fehlertyp (Gier, Angst, Hoffnung, Impuls, Disziplin). Am besten zuerst einrichten.
- **Neuer Eintrag:** Symbol, Datum, P&L, Stimmung vor und nach dem Trade (Skala 1–5), Motivation (System, FOMO, Revenge, Langeweile, Impuls), Marktlage, Einhaltung jeder Regel, Versuchungen und die Lektion, die du mitnimmst.
- **Einträge:** Kacheln (Anzahl, P&L, Trefferquote, Anteil mit Versuchung) und je Eintrag eine Zeile mit Motivation, Anzahl der Versuchungen (⚡), der Regel-Einhaltung als Punkte (grün = eingehalten, rot = gebrochen), P&L und Lektion.



Trade-Psychologie

[+ Neuer Eintrag](#)EINTRÄGE
27GESAMT P&L
1372,57 \$WIN-RATE
63%MIT VERSUCHUNG
48%

Symbol

SPY, AAPL...

Motivation

Alle ▾

Pro Seite

25 ▾

24.10.2025 GE System ●●●●●

+1205,00 \$

Plan eingehalten, keine Überraschungen — genau so soll es laufen.

17.10.2025 ON System ●●●●●

+27,00 \$

Saubere nach System gehandelt, Ergebnis war zweitrangig.

17.10.2025 ON System ●●●●●

+27,00 \$

Saubere nach System gehandelt, Ergebnis war zweitrangig.

13.10.2025 GE FOMO ⚡ 4 ●●●●●

-36,00 \$

FOMO beim Entry gespürt, Positionsgröße im Nachhinein zu groß gewählt.

13.10.2025 GE System ●●●●●

+19,00 \$

Diszipliniert exekutiert, kein Grund etwas zu ändern.

26.09.2025 ON Impuls ⚡ 3 ●●●●●

+37,00 \$

Impulsiv gehandelt, Regelwerk im Kopf übersprungen.

23.09.2025 SPY Impuls ⚡ 4 ●●●●●

-80,00 \$

Kein Fehler im Ergebnis, aber der Prozess war impulsiv — Prozess zählt mehr als Ergebnis.

Psychologie: Einträge

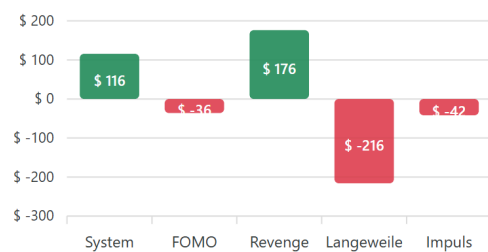


- **Auswertung:** vier Diagramme machen Muster sichtbar — *Cost of Emotion* (Ø P&L je Motivation), *Emotional Sweet Spot* (Ø P&L je Stimmung), *Compliance-Performance* (Ø P&L je Grad der Regeler Einhaltung) und *Regelversuchungen* (wie oft welche Regel in Versuchung führte).

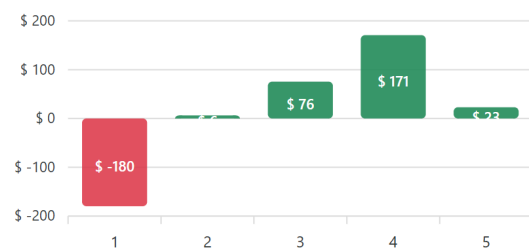
Psychologie — Auswertung

[+ Neuer Eintrag](#)ERFASST
27GESAMT P&L
+1372,57
\$WIN-RATE
63%Ø COMPLIANCE
71%VERSUCHUNGSRATE
48%

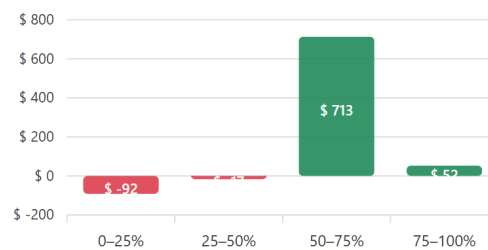
⚡ Cost of Emotion Ø P&L pro Motivation



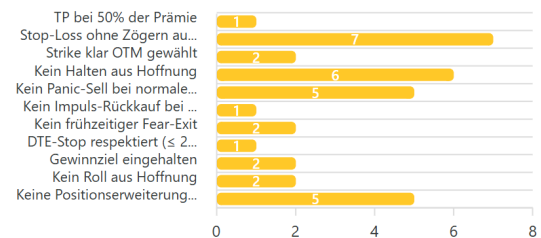
😊 Emotional Sweet Spot Ø P&L pro Gemütszustand (1-5)



🛡️ Compliance-Performance Ø P&L pro Regelerfüllungs-Bucket



⚠️ Regelversuchungen Häufigkeit pro Regel



Psychologie: Auswertung

9. Analyse

Fünf Ansichten zum Tiefergraben:



Ansicht	Was sie zeigt
Symbole	alle je gehandelten Symbole mit Anzahl Kampagnen, letzter Aktivität, Brutto-/Netto-P&L und Trefferquote; Detailseite mit P&L-Verlauf und allen Kampagnen des Symbols
Stats	Ergebnisse nach Tagen, Wochen, Monaten oder Jahren (siehe unten)
Aktivität	wochenweise blätterbare Buchungshistorie über alle Symbole — wie die Transaktions-Historie einer Kampagne, nur für alles
FX-Trades	reine Währungstausche (z. B. USD ↔ EUR); Teilausführungen einer Order werden zu einer Summenzeile zusammengefasst und lassen sich aufklappen
Raw Data	die Trades genau so, wie sie aus dem IBKR-Bericht kamen, mit Verarbeitungsstatus — um eine Zahl bis zu ihrem Ursprung zurückzuverfolgen

21 Tage	13 Wochen	6 Monate	3 Jahre	« 1 Jahr zurück	Aktuell	Vor »
ZEITRAUM	PNL	PROFIT FACTOR	PROCEEDS	COMMISSIONS	NET CASH	BUCHUNGEN
2026 Aktuell	87,00 \$ netto: 77,15 \$ (-5.348,96)	1.02	-55,00 \$	-9,85 \$	435,15 \$	13
2025	5.435,96 \$ netto: 5.085,94 \$ (+5.220,93)	1.19	12.588,58 \$	-350,02 \$	12.664,56 \$	388
2024	215,04 \$ netto: 186,35 \$ (+215,04)	2.09	-74,96 \$	-28,69 \$	-103,65 \$	41
GESAMT (3 Jahre)	5.738,00 \$ netto: 5.349,44 \$		12.458,62 \$	-388,56 \$	12.996,07 \$	442

Stats in der Jahresansicht

Stats schaltet oben zwischen 21 Tagen, 13 Wochen, 6 Monaten und 3 Jahren um und blättert rechts zeitlich vor und zurück; unten steht jeweils die Summe.

- **Wochen, Monate, Jahre** (Bild): P&L (brutto, darunter netto und die Veränderung), Profit Factor, Erlöse (Proceeds), Kommissionen, Cashflow (Net Cash) und Anzahl Buchungen.
- **Tage**: P&L mit Veränderung zum Vortag, Buchungen, Gewinner/Verlierer, Ø Trade, Kommission und der im Monat aufgelaufene P&L.

Tipp

Stats zählt **Buchungen**, nicht Trades: Ein geschlossener Trade besteht aus Eröffnung und Schließung, also zwei Buchungen.

IBKR einrichten — Flex Queries und Kontoauszug

Damit das Journal Daten bekommt, muss im IBKR-Kundenportal einmalig festgelegt werden, **welche Berichte** IBKR erstellt und **wie** sie zugestellt werden. Dieses Kapitel zeigt die Einrichtung Schritt für Schritt — mit den Einstellungen, die im täglichen Betrieb funktionieren.

Insgesamt werden bis zu drei Berichte eingerichtet:

Bericht	Zweck im Journal	Zustellung	Pflicht?
Flex Query für E-Mail	alle Trades, Verfall/Zuteilung von Optionen, Splits	täglich per E-Mail	ja
Flex Query für Web-API	dieselben Daten über einen zweiten, unabhängigen Kanal	Abruf mit Prüfcode	optional
Kontoauszug (CSV)	Kontostand, Cash, Tagesrendite (TWRR) fürs Dashboard	täglich per E-Mail	optional

Hinweis

Die Bilder stammen aus dem deutschsprachigen Kundenportal (Stand Oktober 2026, Paper-Konto). IBKR ändert die Oberfläche gelegentlich — Bezeichnungen und Anordnung können abweichen. Kontonummern, Query-IDs und Prüfcodes sind in allen Bildern unkenntlich gemacht.

Wichtig

Die Einrichtung gilt **je Konto**. Wer mehrere IBKR-Konten nutzt (z. B. Echtgeld, Paper, separates Konto für automatisierten Handel), wiederholt die Schritte im Kundenportal des jeweiligen Kontos — das Paper-Konto hat einen eigenen Login.

1. Wo die Einstellungen liegen

Im Kundenportal oben im Menü **Performance & Berichte** wählen und dort den Reiter **Flex Queries** öffnen.



Übersicht Flex Queries

Nr.	Element	Wofür
1	Kontoumsatz-Flex-Query mit +	neue Flex Query anlegen (Abschnitt 2)
2	🔍 vor einer Query	Detailansicht mit der Query-ID (Abschnitt 3)
3	Zahnrad bei Flex-Queries-Zustellung	tägliche Zustellung per E-Mail (Abschnitt 4)
4	Zahnrad bei Flex-Web-Service-Konfiguration	Prüfcode für den Abruf per Web-API (Abschnitt 5)

Der Reiter **Kontoauszüge** links daneben wird in Abschnitt 6 gebraucht.

2. Flex Query für die Trades anlegen

Über das + bei *Kontoumsatz-Flex-Query* eine neue Query anlegen (auf Englisch heißt sie *Activity Flex Query*). Der Name ist frei wählbar, z. B. `money_report`.

Tipp

Am besten das **Konto bzw. Environment in den Namen** aufnehmen, z. B. `money_report`, `paper_report`, `autom_money_report`. Der Name erscheint im Dateinamen des Mail-Anhangs (`flex.<Query-ID>.money_report.<Datum>.<Datum>.xml`) und im Bericht selbst. So ist auf einen Blick erkennbar, von welchem Konto ein Bericht stammt — die Kontonummer steht zwar auch darin, aber die hat kaum jemand im Kopf. Für den Import spielt der Name keine Rolle: Die Zuordnung läuft immer über die Kontonummer (`accounts.toml`).

Abschnitte — genau diese drei auswählen (aktivierte Abschnitte erscheinen blau mit Haken):

- Trades



- Optionsausübungen, -zuteilungen und -fälligkeiten
- Kapitalmaßnahmen

Einzahlungen mit Haltefrist	Performance-Übersicht seit Monats-/Jahresbeginn in der Basiswährung
FDIC-versicherte Einlagen nach Bank	Positionen des vorangegangenen Zeitraums
Forex-G&V-Details	Provisionsdetails
Fremdwährungssalden	Provisionsgutschriften
Geliehene/verliehene Wertpapiere	Realisierte und unrealisierte Performance-Zusammenfassung in der Basiswährung
Geliehene/verliehene Wertpapiere - Aktivität	Routing-Provisionen
Geliehene/verliehene Wertpapiere - Gebührendetails	Trades ✓
Gewährung von Bezugsrechten	Transaktionsgebühren
HK IPO - Offene Zeichnungen	Transfers
HK IPO - Zeichnungsaktivität	Umsatzsteuer - Details
IBG-Notes	

Abschnitt Trades

Ausstehende Ausübungen	Kontoinformationen
Bartransaktionen	Mark-to-Market-Performancezusammenfassung in der Basiswährung
Bei IBSS gehaltene Wertpapier-Sicherheit	Mutual Fund Dividend Details
CFD-Gebühren	Nettovermögenswert (NAV) in der Basiswährung
Cash-Bericht	Nicht abgewickelte Transfers
Debit-Card-Kontoumsätze	Nicht gebuchte Trades
Details zu Leihgebühren	Offene Positionen
Details zum aufgelaufenen Incentive Coupon	Offene aufgelaufene Dividenden
Eingehende/ausgehende Trade-Transfers	Optionsausübungen, -zuteilungen und -fälligkeiten ✓
Einzahlungen mit Haltefrist	Performance-Übersicht seit Monats-/Jahresbeginn in der Basiswährung

Abschnitt Optionsausübungen, -zuteilungen und -fälligkeiten

IBG-Notes	Umsatzsteuer - Details
Informationen zum Finanzinstrument	Veränderung der aufgelaufenen Dividenden
Kapitalflussrechnung (Statement of Funds)	Veränderung des NAV
Kapitalmaßnahmen ✓	Zinsdetails (Staffelung)
Übersicht zu Positionswertveränderungen	Übersicht zu Netto-Aktienpositionen

Filter

Symbole + Symbol hinzufügen

Derivate einbeziehen

Abschnitt Kapitalmaßnahmen

**Achtung**

Ohne *Optionsausübungen*, *-zuteilungen* und *-fälligkeiten* bleiben verfallene oder zugeteilte Optionen im Journal dauerhaft offen; ohne *Kapitalmaßnahmen* werden Aktiensplits nicht berücksichtigt.

Beim Anklicken eines Abschnitts öffnet sich ein Fenster mit **Optionen** und der **Feldauswahl**. In allen drei Abschnitten „**Alle auswählen**“ anhängen — dann fehlt garantiert kein Feld, das der Import braucht.

Trades: unter *Optionen* nur **Ausführung** (1), dann **Alle auswählen** (2).

Trades

Optionen

Symbolzusammenfassung

Anlageklasse

Order

Ausführung ✓

Geschlossene Lots

Wash Sales

2 ✓ Alle auswählen

✓ Account ID

✓ Account Alias

✓ Model

✓ Currency

✓ FX Rate To Base

✓ Asset Class

✓ Sub Category

✓ Symbol

✓ Description

Feldauswahl Trades

Optionsausübungen, -zuteilungen und -fälligkeiten: keine Optionen, nur **Alle auswählen**.



Optionsausübungen, -zuteilungen und -fälligkeiten

☒	Alle auswählen	
☒	Account ID	≡
☒	Account Alias	≡
☒	Model	≡
☒	Currency	≡
☒	FX Rate To Base	≡
☒	Asset Class	≡
☒	Sub Category	≡
☒	Symbol	≡
☒	Description	≡

Feldauswahl Optionseignisse

Kapitalmaßnahmen: unter *Optionen* nur **Detail** (1), dann **Alle auswählen** (2).

Kapitalmaßnahmen

Optionen

<input type="text" value="Gesamtübersicht"/>	<input type="text" value="Geschlossene Lots"/>	
1 <input checked="" type="text" value="Detail"/>	<input type="text" value="Wash Sales"/>	
2 ☒	Alle auswählen	
☒	Account ID	≡
☒	Account Alias	≡
☒	Model	≡
☒	Currency	≡
☒	FX Rate To Base	≡
☒	Asset Class	≡
☒	Sub Category	≡

Feldauswahl Kapitalmaßnahmen

Zustellungskonfiguration — weiter unten im Formular:



Einstellung	Wert
Format	XML
Zeitraum	Letzte n Kalendertage
Anzahl der Tage	8 (oder mehr, siehe unten)

Zustellungskonfiguration

Der Zeitraum von acht Tagen ist bewusst überlappend: Fällt ein Bericht aus (Feiertag, Mail-Problem), holt der nächste die fehlenden Tage nach. Doppelt gelieferte Trades erkennt der Import und überspringt sie. Ein ausgeschalteter Rechner ist dagegen kein Ausfall: Die Mails warten im Postfach, bis der nächste Lauf sie abholt (siehe Alltag mit dem TradingJournal, Abschnitt 1).

Wer auf der sicheren Seite sein will, kann den Zeitraum auch größer wählen, z. B. **30 Tage**. Dann überbrückt jeder Bericht entsprechend längere Ausfälle bei IBKR oder beim Mailversand. Der Preis: Jeder Bericht enthält mehr Tage und Trades, es werden mehr Daten erzeugt und übertragen, und die Ablageordner der eingelesenen Berichte (`processed` , `deleted`) wachsen schneller, weil jede Datei größer ist. Die Sammeldatei (Summary) wächst dagegen nicht stärker: IBKR liefert je Handelstag einen eigenen Teilbericht, und die Sammeldatei übernimmt jeden Tag nur einmal.

Für Trades **vor** der Einrichtung reicht der tägliche Bericht nicht. Sie werden einmalig mit derselben Query und einem händisch gesetzten Zeitraum abgerufen — siehe TradingJournal-Weaver installieren, Abschnitt 8.

**Allgemeine Konfiguration:**

Einstellung	Wert
Datumsformat	dd/MM/yyyy
Zeitformat	HHmmss
Datum/Uhrzeit-Trennzeichen	; (Semikolon)
Gewinn und Verlust	Standard
Include Offsetting Trade/Cancel Pairs?	Nein
Wechselkurse miteinbeziehen?	Nein
Prüfpfadfelder einbeziehen?	Ja
Konto-Pseudonym anstelle der Konto-ID anzeigen?	Nein
Aufschlüsselung nach Tagen?	Ja

Allgemeine Konfiguration

Datumsformat: dd/MM/yyyy

Zeitformat: HHmmss

Datum/Uhrzeit-Trennzeichen: ; (Semikolon)

Gewinn und Verlust: Standard

Include Offsetting Trade/Cancel Pairs?

Wechselkurse miteinbeziehen?

Prüfpfadfelder einbeziehen?

Konto-Pseudonym anstelle der Konto-ID anzeigen?

Allgemeine Konfiguration: Datum und Uhrzeit

Wichtig

Datumsformat, Zeitformat und Trennzeichen sind **Pflicht** in genau dieser Kombination (23/09/2025 ; 102924). Mit einem anderen Format kann der Import Datum und Uhrzeit der Trades nicht lesen.



Include Offsetting Trade/Cancel Pairs?	<input type="button" value="Ja"/> <input type="button" value="Nein"/>
Wechselkurse miteinbeziehen?	<input type="button" value="Ja"/> <input type="button" value="Nein"/>
Prüfpfadfelder einbeziehen? ⓘ	<input type="button" value="Ja"/> <input type="button" value="Nein"/>
Konto-Pseudonym anstelle der Konto-ID anzeigen?	<input type="button" value="Ja"/> <input type="button" value="Nein"/>
Aufschlüsselung nach Tagen?	<input type="button" value="Ja"/> <input type="button" value="Nein"/>

Weiter

Konto-Pseudonym: Nein

Das Konto-Pseudonym muss auf **Nein** stehen: Der Import ordnet jeden Bericht über die Kontonummer (die letzten vier Ziffern) einem Benutzer und Environment zu.

Mit **Weiter** erscheint eine Prüfseite mit allen Einstellungen. Dort mit **Speichern** (bei einer bestehenden Query: **Änderungen speichern**) abschließen.

Konto-Pseudonym anstelle der Konto-ID anzeigen?	Nein
Aufschlüsselung nach Tagen?	Ja
Datumsformat	dd/MM/yyyy
Zeitformat	HH:mm:ss
Datum/Uhrzeit-Trennzeichen	: (Semikolon)

Zurück

Änderungen speichern

Prüfseite mit Speichern

3. Query-ID finden

Jede Flex Query hat eine Nummer, die **Query-ID**. Sie wird für den Abruf per Web-API gebraucht (Abschnitt 5). Zu finden ist sie über das ⓘ vor der Query in der Übersicht — die Detailansicht zeigt sie ganz oben unter *Details zu Kontoumsatz-Flex-Queries*.



paper_30_report

DU.....



Flex Queries / paper_30_report

Details zu Kontoumsatz-Flex-Queries

Query-ID

Query-Name

paper_30_report

Abschnitte (Bitte mehrere auswählen)

Aufgelaufene Zinsen

Komplexe Positionen

Ausstehende Ausübungen

Kontoinformationen

Query-ID in der Detailansicht

4. Tägliche Zustellung per E-Mail

In der Übersicht beim Kasten **Flex-Queries-Zustellung** auf das Zahnrad klicken (Abschnitt 1, Nr. 3) und auf der folgenden Seite beim Abschnitt **Zustellung von Umsatz-Flex-Queries** erneut auf das Zahnrad.

- **Zustellungsmethode:** E-Mail (1)
- **Verschlüsselung:** Nein
- **Berichte:** die Query aus Abschnitt 2 anhaken, Format **XML** (2)

Zustellungsmethode 1 E-Mail

Verschlüsselung

Suche

	Berichte	Konten	Format
☐	act_paper_apl8	DU.....	XML
2 ☒	paper_30_report	DU.....	XML
☐	test_31_report	DU.....	XML
☐	test_konto	DU.....	XML

Zustellung der Flex Query bearbeiten

Mit **Weiter** und **Speichern** abschließen. Danach schickt IBKR den Bericht täglich per E-Mail.

Wann die Mails kommen — Die Berichte zu einem Handelstag kommen **am folgenden Tag am Nachmittag** (mitteleuropäische Zeit, meist zwischen etwa 14 und 16 Uhr, einzelne Mails auch später). Am **Samstag werden keine Mails verschickt**: Der Bericht für den Handelstag Freitag kommt erst am **Montag**. Wer morgens mit den Daten vom Vortag arbeiten möchte, nutzt zusätzlich den Abruf per Web-API (Abschnitt 5): Dort stehen sie deutlich früher bereit.



So sieht die Mail aus — Betreff *Activity Flex for 10/14/2026*: Das Datum ist der Handelstag im US-Format (Monat/Tag/Jahr), nicht der Versandtag. Der Name der Flex Query steht **nicht** im Betreff, sondern nur im Dateinamen des Anhangs. Die Mails zum Kontoauszug (Abschnitt 6) heißen *Persönlicher Kontoauszug für ...* bzw. *Customized Activity Statement for ...*.

★ ①	Activity Flex for 10/14/2026	• Interactive Brokers Client Services <donotreply@int...	📧 15.10.2026, 14:02
★ ①	Activity Flex for 10/14/2026	• Interactive Brokers Client Services <donotreply@int...	📧 15.10.2026, 14:12
★ ①	Persönlicher Kontoauszug für 10/14/2026	• Interactive Brokers Client Services <donotreply@int...	📧 15.10.2026, 14:51
★ ①	Activity Flex for 10/14/2026	• Interactive Brokers Client Services <donotreply@int...	📧 15.10.2026, 15:14
★ ①	Activity Flex for 10/14/2026	• Interactive Brokers Client Services <donotreply@int...	📧 15.10.2026, 15:55
★ ①	Customized Activity Statement for 10/14/2026	• Interactive Brokers Client Services <donotreply@int...	📧 15.10.2026, 16:04
★ ①	Persönlicher Kontoauszug für 10/14/2026	• Interactive Brokers Client Services <donotreply@int...	📧 15.10.2026, 16:04
★ ①	Persönlicher Kontoauszug für 10/14/2026	• Interactive Brokers Client Services <donotreply@int...	📧 15.10.2026, 18:20
★ ①	Activity Flex for 10/15/2026	• Interactive Brokers Client Services <donotreply@int...	📧 16.10.2026, 14:10
★ ①	Activity Flex for 10/15/2026	• Interactive Brokers Client Services <donotreply@int...	📧 16.10.2026, 14:15
★ ①	Persönlicher Kontoauszug für 10/15/2026	• Interactive Brokers Client Services <donotreply@int...	📧 16.10.2026, 14:25
★ ①	Activity Flex for 10/15/2026	• Interactive Brokers Client Services <donotreply@int...	📧 16.10.2026, 15:18
★ ①	Customized Activity Statement for 10/15/2026	• Interactive Brokers Client Services <donotreply@int...	📧 16.10.2026, 15:31
★ ①	Persönlicher Kontoauszug für 10/15/2026	• Interactive Brokers Client Services <donotreply@int...	📧 16.10.2026, 15:31
★ ①	Activity Flex for 10/15/2026	• Interactive Brokers Client Services <donotreply@int...	📧 16.10.2026, 15:53

Posteingang: Die Berichte zum 14. und 15.10. kommen jeweils am Folgetag am Nachmittag

Achtung

Der Abruf in **TradingJournal-Weaver** **löscht die verarbeiteten IBKR-Mails endgültig aus dem Posteingang** — die Anhänge liegen danach in `C:\Trade_XML\...` bzw. `C:\Trade_CSV\...`. Das betrifft alle Mails von `donotreply@interactivebrokers.com`, deren Betreff *Activity Flex*, *Trade Confirmation Flex*, *Activity Statement* oder *Kontoauszug* enthält — auch solche ohne verwertbaren Anhang. Wer diese Mails zusätzlich aufbewahren möchte, richtet im Mailprogramm eine Kopie bzw. Weiterleitung in einen anderen Ordner ein.

Achtung

IBKR schickt die Berichte **immer an die E-Mail-Adresse, die beim Anlegen des Kontos hinterlegt wurde** — eine eigene Empfängeradresse nur für Berichte gibt es nicht. Das Postfach, das der Abruf in **TradingJournal-Weaver** liest (`IMAP_USERNAME`), muss also diese Adresse sein oder die Mails dorthin weiterleiten. Bei einer Weiterleitung muss der Absender `donotreply@interactivebrokers.com` erhalten bleiben — der Abruf filtert danach.

5. Abruf per Web-API (optional)

Der Abruf per Web-API ist ein zweiter, unabhängiger Weg zu denselben Daten — nützlich, falls an einem Tag keine Mail ankommt. Und er ist **schneller**: IBKR erstellt den Bericht erst bei der Anfrage, die Daten des Vortags stehen deshalb in der Regel schon **am Morgen** bereit — auch am Samstag für den



Handelstag Freitag, an dem keine Mail kommt. Ist der Bericht bei IBKR gerade noch in Arbeit, wartet der Abruf kurz und versucht es automatisch erneut. Dafür braucht es zwei Dinge: eine eigene Flex Query und einen Prüfcode.

Eigene Query: eine zweite Flex Query anlegen, **mit genau denselben Einstellungen** wie in Abschnitt 2 (z. B. Name `act_money_api`). Diese Query wird **nicht** für die E-Mail-Zustellung angehakt — sie wird nur abgerufen. Ihre Query-ID wie in Abschnitt 3 ablesen.

Prüfcode: In der Übersicht beim Kasten **Flex-Web-Service-Konfiguration** auf das Zahnrad klicken (Abschnitt 1, Nr. 4). IBKR nennt den Zugangsschlüssel **Prüfcode** (englisch *Token*).

Flex-Web-Service: Prüfcode

Nr.	Element	Bedeutung
—	Flex-Web-Service-Status	muss angehakt sein
1	Aktiver Prüfcode	der Schlüssel für <code>TradingJournal-Weaver</code>
2	Aktivierungszeitraum	Gültigkeit — rund ein Jahr
3	Neuen Prüfcode generieren	erzeugt einen neuen Schlüssel, der alte wird sofort ungültig

Gibt es noch keinen Prüfcode, wird er mit *Neuen Prüfcode generieren* erstellt. In `TradingJournal-Weaver/.env` werden Prüfcode und Query-ID als Paar eingetragen:

```
IBKR_FLEX_QUERIES=Prüfcode:Query-ID
```

Mehrere Konten bzw. Queries werden durch Kommas getrennt
(`Prüfcode1:QueryID1,Prüfcode2:QueryID2`).

**Achtung**

Der Prüfcode **läuft nach rund einem Jahr ab**. Danach scheitert der Web-API-Abruf, bis ein neuer Prüfcode erzeugt und in der `.env` eingetragen ist — das Ablaufdatum am besten im Kalender notieren. *Neuen Prüfcode generieren* nur dann klicken, wenn der alte abgelaufen oder kompromittiert ist. Außerdem: Viele fehlgeschlagene Abrufe kurz hintereinander können zu einer vorübergehenden Sperre durch IBKR führen.

Vorsicht

Der Prüfcode ist ein Zugangsschlüssel zu den Kontodaten. Er gehört nur in die `.env` (die nie committet wird) — nicht in Screenshots, Chats oder Support-Anfragen.

6. Kontoauszug (CSV) für Kontostand und Rendite

Die Tageswerte des Kontos — Nettovermögen, Cash, Tagesrendite — liefert IBKR **nicht** über Flex Queries, sondern nur im Kontoauszug, und den **nur per E-Mail**.

Hinweis

Der Kontoauszug ist optional — je Konto frei wählbar. Ohne ihn funktioniert das Journal vollständig: Trades, Kampagnen, Gewinn/Verlust und Statistiken kommen alle aus der Flex Query. Es fehlen nur die Kontostands-Kacheln auf dem Dashboard: *NAV aktuell*, *NAV Änderung (Tag)*, *TWRR*, *Cash EUR*, *Cash USD*, *Realisiert (Tag)*, *Unrealisiert (Tag)* und der *NAV-Verlauf* werden dann ausgeblendet, *Max Drawdown* zeigt „Keine NAV-Daten vorhanden“. Das Portfolio selbst zeigt die IBKR-Weboberfläche ohnehin aktueller und ausführlicher — wer dort nachsieht, kann auf den Kontoauszug gut verzichten.

Wichtig

Dafür muss ein **eigener, benutzerdefinierter Kontoauszug** angelegt werden. Die Standard-Kontoauszüge von IBKR lassen sich nicht täglich per E-Mail zustellen — nur ein selbst angelegter Kontoauszug kommt jeden Tag als CSV ins Postfach.

Im Menü **Performance & Berichte** den Reiter **Kontoauszüge** öffnen. Unten im Abschnitt **Benutzerdefinierte Kontoauszüge** einen neuen Kontoauszug anlegen, Kontoauszugstyp **Kontoauszug**.

Abschnitte — diese fünf werden vom Import gelesen und sind Pflicht:



- Kontoinformationen
- Nettovermögenswert
- Cash-Bericht
- Übersicht: realisierter & unrealisierter G&V
- Übersicht Monats- & Jahresbeginn bis heute

Weitere Abschnitte stören nicht. Die Bilder zeigen eine funktionierende Auswahl, die zusätzlich u. a. *Trades*, *Kapitalmaßnahmen*, *MTM-Performance-Übersicht* und *Transaktionsgebühren* enthält.

Standardabschnitte

Alle	Kontoinformationen ✓
Anpassungen	Kundengebühren
Aufgelaufene Zinsen	Leih-/Darlehensgebühren
Ausgehende Trade-Transfers	Leihen/Darlehen
Barausgleich bei Optionen ✓	MTM-Performance-Übersicht ✓
Bill Payment	Nettovermögenswert ✓
Cash-Bericht ✓	Nicht abgewickelte Transfers
Debit-Card-Kontoumsätze	Nicht gebuchte Trades
Details zu Leihgebühren	Offene Positionen
Eingehende Trade-Transfers	Provisionsgutschriften

Kontoauszug: Abschnitte (1)

Eingehende Trade-Transfers	Provisionsgutschriften
Einzahlungen mit Haltefrist	Quellensteuer
Emissionsgutschriften	Rechtshinweise
Fremdwährungssalden	Routing-Provisionen
Gewährung von Bezugsrechten	Soft-Dollar-Aktivitäten
HK IPO - Offene Zeichnungen	Sonstiges Einkommen
HK IPO - Zeichnungsaktivität	Trades ✓
Informationen zum Finanzinstrument	Transaktionscodes
Kapitalmaßnahmen ✓	Transaktionsgebühren ✓
Kombinierte Dividenden	Transfers
Kombinierte Ein-/Auszahlungen	Veränderung der aufgelaufenen Dividenden

Kontoauszug: Abschnitte (2)

Einstellungen und Zustellungskonfiguration am Ende des Formulars:



Einstellung	Wert
Konto-ID durch Konto-Pseudonym ersetzen?	Nein (1)
Format	CSV (2)
Zeitraum	Täglich (2)
Sprache	English (2)

Kontoauszug: Format, Zeitraum, Sprache

Wichtig

Die **Sprache muss English sein**: Der Import erkennt die Abschnitte und Zeilen an ihren englischen Bezeichnungen (z. B. *Net Asset Value*, *Ending Cash*).

Hinweis

Im Bild steht das Pseudonym (1) auf *Ja*. Das funktioniert nur, solange für das Konto kein Pseudonym vergeben ist — dann setzt IBKR die teilweise maskierte Kontonummer ein. **Nein** ist die sichere Wahl.

Zustellung: zurück im Reiter **Kontoauszüge** rechts beim Kasten **Zustellung der Kontoauszüge** auf das Zahnrad klicken.



Kontoauszüge

Flex Querles Steuerdokumente Drittanbieter-Berichte Transaktionsverlauf

Standardkontoauszüge

- Kontoauszug**
Liefert Informationen zu Beständen, Transaktionen, Salden und zur Wertentwicklung, einschließlich einer Aufschlüsselung Ihrer Wertpapier- und Futures-Segmente, falls vorhanden.
- MTM-Übersicht**
Liefert eine Übersicht über Ihr Konto mit Fokus auf Ihrem Mark-to-Market-G&V.
- Übersicht: realisierter G&V**
Liefert eine Übersicht über die realisierten und unrealisierten Gewinne und Verluste Ihres Depots.
- Aufsichtsrechtliche Meldung zu Rohstoffen/Terminkontrakten**
Liefert eine Übersicht mit Schwerpunkt auf Ihrem Rohstoff-Segment.
- Handelsbestätigung**
Listet die Trades des aktuellen Tages sowie vergangene Trades in Ihrem Depot auf.
- Transaktionsaufteilung**
Listet die Trades auf, die am heutigen Tag und in der Vergangenheit von Ihrem Depot aus anderen zugewiesen wurden.

Zustellung der Kontoauszüge

- Monatlicher Kontoauszug
- ☒ paper_act

Kontoauszüge: Zustellung

Im Abschnitt **Zustellung von täglichen, benutzerdefinierten Auszügen** über das Zahnrad Zustellungsmethode **E-Mail** wählen und den neuen Kontoauszug mit Format **CSV** anhängen — wie bei der Flex Query in Abschnitt 4.

Zustellung von täglichen, benutzerdefinierten Auszügen

Zustellungsmethode: E-Mail
Verschlüsselung: Nein

Berichte	Konten	Format
paper_act	DU.....	CSV

Zustellung von monatlichen, benutzerdefinierten Auszügen

Es wurden keine Berichte für die Zustellung konfiguriert.

Zustellung der täglichen, benutzerdefinierten Auszüge

Die Mail kommt an dieselbe Konto-Adresse wie die Flex Query (Betreff *Activity Statement* bzw. *Kontoauszug*). Die Standard-Kontoauszüge oben auf der Seite (täglich, monatlich) sind für das Journal nicht nötig. Der **monatliche Kontoauszug lässt sich in der Zustellung allerdings nicht abwählen** — er kommt also zusätzlich einmal im Monat per Mail, auch wenn er nicht gebraucht wird. Enthält sein Betreff *Kontoauszug* bzw. *Activity Statement*, löscht der Abruf auch diese Mail aus dem Posteingang (siehe Hinweis in Abschnitt 4) — wer den Monatsauszug behalten möchte, sichert ihn vorher oder lädt ihn später im IBKR-Kundenportal herunter.



7. Zusammenfassung: was wohin gehört

Aus dem Kundenportal	Eintrag in <code>TradingJournal-Weaver</code>
E-Mail-Adresse des Kontos (bzw. Postfach mit Weiterleitung)	<code>.env</code> : <code>IMAP_USERNAME</code> , <code>IMAP_PASSWORD</code> , <code>IMAP_SERVER</code>
Prüfcode + Query-ID der Web-API-Query	<code>.env</code> : <code>IBKR_FLEX_QUERIES=Prüfcode:Query-ID</code>
Kontonummer (letzte vier Ziffern)	<code>accounts.toml</code> : Zuordnung zu Benutzer und Environment

Checkliste je Konto:

1. Flex Query mit *Trades, Optionsausübungen, -zuteilungen und -fälligkeiten, Kapitalmaßnahmen* — jeweils *Alle auswählen*
2. XML, letzte 8 Kalendertage (oder mehr), `dd/MM/yyyy` · `HHmmss` · `;` , Pseudonym *Nein*
3. Flex Query für die tägliche E-Mail-Zustellung angehakt
4. (*optional*) zweite, gleich eingestellte Query + Prüfcode für die Web-API, Ablaufdatum notiert
5. (*optional*) Benutzerdefinierter Kontoauszug: fünf Pflicht-Abschnitte, CSV, täglich, English, Pseudonym *Nein*
6. (*optional*) Kontoauszug für die tägliche E-Mail-Zustellung angehakt
7. Kontonummer in `accounts.toml` eingetragen

Windows vorbereiten

Diese Anleitung erklärt Schritt für Schritt, welche Programme auf einem Windows-Rechner installiert werden müssen, damit das Trading-Journal-Projekt läuft — **auch wenn du bisher wenig Erfahrung mit Programmieren, Terminals oder Entwickler-Tools hast.**

Es gibt zu diesem Thema auch eine kürzere, technischere Version im technischen Handbuch: Installation: Windows-Grundausrüstung. Diese Seite hier erklärt dieselben Schritte, nur ausführlicher und mit mehr Erklärungen dazwischen. Am Ende landest du beim gleichen Ergebnis.

Zeitaufwand: ca. 30–60 Minuten, je nach Internetverbindung (mehrere Programme müssen heruntergeladen werden).

Inhalt

1. Was du am Ende hast
2. Ein paar Begriffe vorab
3. Git installieren
4. XAMPP installieren (Datenbank + Webserver)
5. Composer installieren
6. Node.js installieren
7. Python installieren
8. Projektordner anlegen und Code herunterladen
9. Alles überprüfen
10. Wie geht's weiter?

1. Was du am Ende hast

Nach dieser Anleitung sind auf dem Rechner fünf Programme installiert:



Programm	Wofür
Git	Lädt den Programmcode von GitHub herunter und hält ihn aktuell
XAMPP	Enthält die Datenbank (MySQL), die Programmiersprache PHP und einen Webserver (Apache, für den Anfang nicht nötig)
Composer	Installiert Zusatz-Bausteine für den PHP-Teil des Projekts
Node.js	Baut die Optik (CSS/JavaScript) der Weboberfläche
Python	Führt die Import-Skripte aus, die die Handelsdaten von Interactive Brokers holen und verarbeiten

Außerdem liegen am Ende zwei Ordner mit dem eigentlichen Projekt-Code auf der Festplatte.

2. Ein paar Begriffe vorab

Diese Begriffe tauchen im Folgenden immer wieder auf — kurz erklärt, damit nichts wie „Fachchinesisch“ wirkt:

- **PowerShell / Terminal:** Ein Fenster, in das man statt mit der Maus zu klicken, Befehle als Text eintippt und mit Enter bestätigt. Wirkt zuerst ungewohnt, ist aber nur eine andere Art, dem Computer zu sagen, was er tun soll. Wie man es öffnet, steht in Schritt 8.
- **Installer:** Eine `.exe`-Datei, die man herunterlädt und doppelklickt — startet einen Installationsassistenten mit „Weiter“-Buttons, wie bei jedem anderen Programm auch.
- **Git / Repository / „klonen“:** Der Programmcode für dieses Projekt liegt online auf [GitHub](#) in zwei separaten „Repositories“ (Projektordnern mit Versionsverlauf). „Klonen“ bedeutet: eine komplette Kopie davon auf den eigenen Rechner herunterladen.
- **PATH:** Eine Liste von Ordnern, in denen Windows automatisch nach Programmen sucht, wenn man im Terminal nur den Programmnamen eintippt (z.B. `php` statt dem kompletten Pfad `C:\xampp\php\php.exe`). Diese Liste muss man XAMPP nach der Installation manuell hinzufügen — das erklärt Schritt 4.

Wichtiger Tipp für die ganze Anleitung: Befehle, die in einem grauen Kasten stehen, immer **exakt so abtippen oder besser kopieren** (Rechtsklick → Kopieren, dann im Terminal-Fenster Rechtsklick → Einfügen). Groß-/Kleinschreibung und Sonderzeichen spielen eine Rolle.

3. Git installieren

1. Diese Seite öffnen: git-scm.com/install/windows und auf „**Click here to download**“ klicken — das lädt den Installer für einen normalen 64-Bit-Windows-Rechner herunter. Der Download startet **nicht** von selbst.



2. Die heruntergeladene Datei doppelklicken, um die Installation zu starten.
3. Durch den Installationsassistenten klicken. Bei der Seite mit der Überschrift „**Adjusting your PATH environment**“ die Option „**Git from the command line and also from 3rd-party software**“ auswählen (meistens schon vorausgewählt).
4. Bei allen anderen Seiten die vorausgewählte Option („Next“) übernehmen, bis die Installation fertig ist.

Für das Herunterladen und spätere Aktualisieren des Codes ist sonst nichts einzurichten — auch kein GitHub-Konto.

Hinweis

Nur wer später selbst Änderungen am Code einreichen möchte, sollte Git noch Namen und Mailadresse mitteilen (PowerShell: `git config --global user.name "Dein Name"` und `git config --global user.email "deine@mail-adresse.at"`). Für die normale Nutzung ist das nicht nötig.

4. XAMPP installieren (Datenbank + Webserver)

XAMPP bringt in einem Paket drei Dinge mit: **MySQL** (die Datenbank, in der alle Handelsdaten gespeichert werden), **PHP** (die Programmiersprache, in der die Weboberfläche geschrieben ist) und **Apache** (ein Webserver). Gebraucht werden zunächst nur MySQL und PHP — die Weboberfläche startet später mit einem eigenen, eingebauten Mini-Webserver. Apache ist nur für eine fortgeschrittene Variante nötig (Kapitel TradingJournal installieren, Abschnitt „Starten und öffnen“).

Wichtig bei der Auswahl: Auf der Download-Seite gibt es mehrere Versionen nebeneinander. Für dieses Projekt wird **PHP Version 8.2 oder neuer** gebraucht — also die „**8.2.x**“-Version (oder eine neuere Zahl) auswählen, nicht blind irgendeine.

Hinweis

Das PHP in XAMPP veraltet mit der Zeit: XAMPP liefert neue PHP-Versionen oft erst Wochen nach einem Sicherheitsupdate aus. Bei einer bekannt gewordenen PHP-Sicherheitslücke lässt sich PHP auch selbst austauschen — siehe Updates einspielen, Abschnitt „Sicherheitsupdates für PHP“.

Installieren

1. Seite öffnen: apachefriends.org
2. Installer für die passende PHP-Version (siehe oben) herunterladen und doppelklicken.
3. Bei der Frage nach den Komponenten mindestens **Apache**, **MySQL** und **PHP** angehakt lassen — den Rest (FileZilla, Mercury Mail, Tomcat) kann man abwählen, wird nicht gebraucht.



4. Als Installationsordner den vorgeschlagenen Pfad `C:\xampp` übernehmen.
5. Installation abschließen, XAMPP am Ende **nicht sofort starten** (Häkchen ggf. entfernen) — der nächste Schritt kümmert sich darum.

XAMPP starten

1. **XAMPP Control Panel** über das Startmenü öffnen.
2. Rechts neben **MySQL** auf **Start** klicken. Die Zeile sollte kurz danach grün hinterlegt sein und „Running“ anzeigen. **Apache** muss nicht gestartet werden.

PHP für das Terminal verfügbar machen

Ohne diesen Schritt kennt das Terminal die Befehle `php` und `composer` nicht.

1. Windows-Taste drücken, `Umgebungsvariablen` eintippen, „**Umgebungsvariablen für dieses Konto bearbeiten**“ öffnen.
2. Im oberen Bereich („Benutzervariablen“) die Zeile `Path` anklicken (einmal, nicht doppelt), dann auf **Bearbeiten** klicken.
3. Auf **Neu** klicken, dann eintippen: `C:\xampp\php`
4. Nochmal auf **Neu** klicken, dann eintippen: `C:\xampp\mysql\bin`
5. Überall auf **OK** klicken, um die Fenster zu schließen.
6. **Wichtig:** Alle offenen PowerShell-Fenster schließen — die Änderung wirkt erst in einem neu geöffneten Fenster.

Testen: Neues PowerShell-Fenster öffnen, `php -v` eintippen und Enter drücken. Es sollte etwas wie `PHP 8.2.x` angezeigt werden. Kommt stattdessen eine Fehlermeldung, Schritt 3–6 nochmal genau durchgehen.

Eine PHP-Einstellung anpassen („zip“-Erweiterung)

Dieser Schritt ist wichtig, aber leicht zu übersehen — XAMPP hat eine Funktion namens `zip` standardmäßig **ausgeschaltet**, die später beim Einrichten der Weboberfläche gebraucht wird. Ohne diesen Schritt kommt dort eine Fehlermeldung mit dem Wort „zip“ darin.

1. Den Datei-Explorer öffnen und zu `C:\xampp\php` navigieren.
2. Die Datei `php.ini` mit einem Text-Editor öffnen (Rechtsklick → **Öffnen mit** → **Editor**).
3. Im Editor mit `Strg+F` nach `zip` suchen. Es sollte eine Zeile erscheinen, die so aussieht:
`;extension=zip`
4. Das Semikolon (`;`) ganz am Anfang dieser einen Zeile löschen, sodass nur noch `extension=zip` dasteht. Sonst nichts an der Zeile verändern.
5. Datei speichern (`Strg+S`) und den Editor schließen.
6. Ein **neues** PowerShell-Fenster öffnen (wichtig: ein neues, kein altes weiterverwenden) und eintippen: `php -m`. In der langen Liste, die erscheint, sollte irgendwo `zip` stehen.



XAMPP automatisch starten lassen (optional, aber empfohlen)

Ohne diesen Schritt muss man vor jeder Nutzung erst das XAMPP Control Panel öffnen und MySQL manuell starten. Mit diesem Schritt startet die Datenbank automatisch mit Windows — das ist auch für den automatischen täglichen Datenimport wichtig, der eine laufende Datenbank braucht.

1. XAMPP Control Panel schließen, dann über das Startmenü erneut öffnen — diesmal per **Rechtsklick** auf das XAMPP-Symbol → „**Als Administrator ausführen**“.
2. Ganz links in der Zeile **MySQL** gibt es eine kleine Spalte „**Svc**“ — dort das Kästchen anklicken.
3. Meist erscheint jetzt ein Fenster der **Benutzerkontensteuerung** von Windows („Möchten Sie zulassen, dass durch diese App Änderungen an Ihrem Gerät vorgenommen werden?“). Das ist normal, weil MySQL dabei als Windows-Dienst eingerichtet wird — mit **Ja** bestätigen.

Falls das nicht klappt: kein Problem, XAMPP funktioniert auch ohne diesen Schritt — dann einfach vor jeder Nutzung kurz das Control Panel öffnen und MySQL starten.

Datenbank-Passwort

Nach einer frischen XAMPP-Installation hat die Datenbank einen Benutzer namens **root** **ohne Passwort**. Für die lokale Nutzung auf dem eigenen Rechner ist das in Ordnung — diese Datenbank ist nicht aus dem Internet erreichbar. Dieser leere Wert wird später beim Einrichten des eigentlichen Projekts (in einer Datei namens **.env**) einfach so übernommen.

5. Composer installieren

Composer lädt automatisch alle Zusatz-Bausteine, die der Weboberflächen-Teil des Projekts braucht.

1. Seite öffnen: getcomposer.org/download
2. Datei **Composer-Setup.exe** herunterladen und doppelklicken.
3. Der Installer erkennt PHP normalerweise automatisch (weil es ja schon installiert ist). Falls nachgefragt wird, wo PHP liegt: **C:\xampp\php\php.exe** eintragen.
4. Restliche Schritte mit „Weiter“ durchklicken.

Testen: Neues PowerShell-Fenster öffnen, **composer -V** eintippen. Es sollte eine Versionsnummer erscheinen.

6. Node.js installieren

Node.js wird gebraucht, um die Optik (Farben, Icons, Layout) der Weboberfläche zusammenzubauen.

1. Seite öffnen: nodejs.org



2. Die als „LTS“ gekennzeichnete Version herunterladen (das ist die empfohlene, stabilste Version) und installieren — überall die Standardeinstellungen übernehmen.

Testen: Neues PowerShell-Fenster öffnen, `node -v` und danach `npm -v` eintippen — beide sollten eine Versionsnummer anzeigen. Bei `node -v` muss sie **mindestens v20.19** sein (die aktuelle LTS-Version erfüllt das); ältere Versionen können die Optik nicht bauen.

7. Python installieren

Python führt die Skripte aus, die die Handelsdaten von Interactive Brokers abrufen und in die Datenbank einsortieren.

Gebraucht wird **Python 3.14** — nicht die neueste Version, die python.org gerade ganz oben anbietet. Mit einer neueren Version (z. B. 3.15) lassen sich einige Zusatz-Pakete, die `TradingJournal-Weaver` braucht, noch nicht installieren.

1. Seite öffnen: python.org/downloads/windows
2. In der Liste den obersten Eintrag mit **Python 3.14** suchen (z. B. *Python 3.14.8 - Sept. 30, 2026*) und dort auf „**Windows installer (64-bit)**“ klicken. Das ist der eigenständige Installer (*standalone installer*).
3. Die heruntergeladene Datei doppelklicken.

Achtung

Nicht den *Python install manager* verwenden, der auf python.org ebenfalls angeboten wird, und auch nicht den großen Knopf *Download Python 3.x* auf der Startseite der Downloads — beide installieren die jeweils neueste Version. Diese Anleitung geht vom eigenständigen Installer für Python 3.14 aus.

4. **Ganz wichtig, leicht zu übersehen:** Ganz unten auf der ersten Installer-Seite gibt es eine Checkbox „**Add python.exe to PATH**“ — die **muss angehakt sein**, bevor man auf „Install Now“ klickt. Ohne diesen Haken funktioniert `python` später nicht im Terminal.
5. Installation abschließen.

Testen: Neues PowerShell-Fenster öffnen, `python --version` eintippen — sollte `Python 3.14.x` anzeigen. Mit dieser Version ist das Projekt getestet.

8. Projektordner anlegen und Code herunterladen



Terminal öffnen und Startordner verstehen

Ein neu geöffnetes PowerShell-Fenster startet **nicht immer am selben Ort** — meistens im eigenen Benutzerordner (`C:\Users\<dein AnmeldeName>`), aber wenn man es über „Als Administrator ausführen“ öffnet, landet man stattdessen in einem Systemordner (`C:\Windows\System32`). Falls du dich also mal wunderst, warum du „irgendwo landest“: das ist normal, kein Fehler.

Man kann jederzeit mit dem Befehl `cd` in einen anderen Ordner wechseln, z.B.:

```
cd C:\Users\<dein AnmeldeName>
```

(`<dein AnmeldeName>` durch den eigenen Windows-Anmeldenamen ersetzen — der steht z.B. oben im Startmenü.)

Projektordner anlegen

Ein neues PowerShell-Fenster öffnen (normal, **nicht** als Administrator) und folgende zwei Zeilen eintippen:

```
New-Item -ItemType Directory -Force -Path C:\Users\<dein AnmeldeName>\trading  
Set-Location C:\Users\<dein AnmeldeName>\trading
```

Die erste Zeile legt einen neuen, leeren Ordner namens `trading` an. Die zweite wechselt dort hinein — alle folgenden Befehle in diesem Fenster beziehen sich jetzt auf diesen Ordner. (Der Name/Ort dieses Ordners ist frei wählbar — `trading` ist nur ein Vorschlag.)

Die zwei Projekte herunterladen („klonen“)

Jetzt die folgenden zwei Befehle **einzeln, nacheinander** eintippen (nach jeder Zeile Enter drücken und warten, bis der Download fertig ist, bevor die nächste Zeile kommt):

```
git clone https://github.com/mike-1140/TradingJournal-Weaver.git
```

```
git clone https://github.com/mike-1140/TradingJournal.git
```

Die Repositories sind öffentlich — eine Anmeldung bei GitHub ist dafür nicht nötig.

Ergebnis: Im Ordner `trading` liegen jetzt zwei Unterordner: `TradingJournal-Weaver` und `TradingJournal`. Das lässt sich mit folgendem Befehl überprüfen — er listet den Inhalt des aktuellen Ordners auf:

```
Get-ChildItem
```



9. Alles überprüfen

Ein **neues** PowerShell-Fenster öffnen (wichtig: neu, nicht dasselbe wie vorher weiterverwenden) und der Reihe nach eintippen — nach jeder Zeile sollte eine Versionsnummer erscheinen, keine Fehlermeldung:

```
git --version  
php -v  
composer -V  
node -v  
npm -v  
python --version
```

Zusätzlich: XAMPP Control Panel öffnen und prüfen, dass **MySQL** grün („Running“) angezeigt wird.

Falls bei einem dieser Befehle eine Fehlermeldung kommt (z.B. „wird nicht als Befehl erkannt“): meistens bedeutet das, dass entweder die Installation noch nicht komplett ist, oder dass die PATH-Änderung (Schritt 4) noch nicht wirkt, weil das Terminal-Fenster zu alt ist — ein frisches Fenster öffnen und nochmal probieren.

10. Wie geht's weiter?

Die Grundausstattung ist fertig. Als Nächstes folgen die eigentlichen Projekt-Einrichtungen — dort geht es dann konkret um dieses Trading-Journal-Projekt (Datenbank befüllen, erste Anmeldung usw.):

1. TradingJournal-Weaver installieren — Einrichtung der Datenimport-Pipeline
2. TradingJournal installieren — Einrichtung der Weboberfläche, ebenfalls für Einsteiger erklärt
3. Erster Start — UI-Erstkonfiguration — erste Schritte in der Weboberfläche



TradingJournal-Weaver installieren

Diese Anleitung richtet die Datenimport-Pipeline ([TradingJournal-Weaver](#)) auf einem Rechner ein — sie holt die Handelsdaten von Interactive Brokers und verarbeitet sie für das Trading-Journal. Schritt für Schritt, mit Erklärungen, auch wenn du bisher wenig Erfahrung mit Terminal-Befehlen oder Python hast. Es gibt dazu auch eine kürzere, technischere Version: Technisches Handbuch: Installation: TradingJournal-Weaver. Beide führen zum selben Ergebnis.

Bevor du hier startest: Die Grundausstattung (Git, Python, XAMPP, ...) muss bereits installiert sein — siehe Windows vorbereiten.

Inhalt

1. Projekt herunterladen
2. Python-Arbeitsumgebung einrichten
3. Konfigurationsdatei anlegen
4. Datenbank-Tabellen anlegen
5. Automatisierung einrichten
6. Ersten Testlauf machen
7. Praktisch: Schnellzugriff einrichten
8. Alte Daten nachholen (einmalig)

1. Projekt herunterladen

Ein PowerShell-Fenster öffnen (Windows-Taste drücken, [PowerShell](#) eintippen, Enter) und in den Ordner wechseln, in dem die anderen Projekte schon liegen bzw. liegen sollen (z.B. `C:\Users\<dein Name>\trading`):

```
cd C:\Users\<dein Name>\trading
```

Falls [TradingJournal-Weaver](#) dort noch nicht liegt, herunterladen („klonen“):

```
git clone https://github.com/mike-1140/TradingJournal-Weaver.git
cd TradingJournal-Weaver
```



2. Python-Arbeitsumgebung einrichten

Python-Projekte arbeiten normalerweise mit einer eigenen, abgeschotteten „Arbeitsumgebung“ (venv — kurz für *virtual environment*) pro Projekt. Das verhindert, dass sich verschiedene Python-Projekte auf demselben Rechner gegenseitig mit unterschiedlichen Versionen von Zusatz-Paketen in die Quere kommen. Drei Befehle, nacheinander:

```
python -m venv .venv
.venv\Scripts\activate
pip install -r requirements.txt
```

Der zweite Befehl „aktiviert“ die Arbeitsumgebung für dieses Terminal-Fenster — am vorangestellten `(.venv)` im Prompt erkennbar. Der dritte lädt alle Zusatz-Pakete herunter, die `TradingJournal-Weaver` braucht (kann ein bis zwei Minuten dauern).

Wichtig: Die Aktivierung (`.venv\Scripts\activate`) gilt nur für das aktuell offene Terminal-Fenster. Bei jedem neuen Fenster muss man diesen einen Befehl wiederholen — dafür gibt es aber eine Abkürzung, siehe Schritt 7.

3. Konfigurationsdatei anlegen

Wie beim Trading-Journal selbst braucht auch `TradingJournal-Weaver` eine `.env`-Datei mit den eigenen Zugangsdaten. Vorlage kopieren:

```
copy .env.example .env
```

Jetzt die neue Datei `.env` im `TradingJournal-Weaver`-Ordner mit einem Text-Editor öffnen (Rechtsklick → **Öffnen mit** → **Editor**). Folgende Werte eintragen bzw. anpassen:



Zeile in <code>.env</code>	Was dort rein muss
<code>DB_USER=</code>	<code>root</code> (XAMPP-Standard)
<code>DB_PASS=</code>	leer lassen, falls kein XAMPP-Datenbank-Passwort vergeben wurde
<code>DEFAULT_USER=</code>	der eigene Name, z.B. <code>anna</code> — dazu gleich mehr
<code>IMAP_USERNAME=</code>	das Postfach, an das IBKR die Berichte schickt (Kapitel IBKR einrichten, Abschnitt 4)
<code>IMAP_PASSWORD=</code>	das Passwort dieses Postfachs
<code>IMAP_SERVER=</code>	der Posteingangs-Server (IMAP) des Mailanbieters, z.B. <code>imap.gmx.com</code>
<code>IBKR_FLEX_QUERIES=</code>	nur für den Abruf per Web-API: Prüfcode und Query-ID als Paar (Kapitel IBKR einrichten, Abschnitt 5) — sonst leer lassen

Was hat es mit `DEFAULT_USER` auf sich? Falls mehrere Personen dasselbe Trading-Journal nutzen (z.B. eine Familie), bekommt jede Person ihre eigenen Datenbank-Tabellen mit einem eigenen „Präfix“ (einer Art Namens-Kürzel vor jedem Tabellennamen, z.B. `anna_trades_raw`). Welches Präfix zu welchem Namen gehört, steht zentral in der Datei `users.toml`. Auch sie wird aus einer Vorlage angelegt (`copy users.toml.example users.toml`) und dann um die eigenen Namen ergänzt — Beispiel unten. `DEFAULT_USER` ist der Hauptbenutzer dieses Rechners — sein Präfix gilt, wenn ein Befehl ohne ausdrückliche Benutzerangabe läuft. Der tägliche automatische Lauf verarbeitet dagegen alle Benutzer, die in `pipeline.toml` stehen (Schritt 5).

Welches IBKR-Konto zu wem gehört, steht in einer dritten Datei, `accounts.toml`. Sie wird ebenfalls aus einer Vorlage angelegt:

```
copy accounts.toml.example accounts.toml
```

Ohne passenden Eintrag dort überspringt der Import die Berichte eines Kontos (Meldung „Unbekannte accountId“). Als Schlüssel dienen die **letzten vier Ziffern der IBKR-Kontonummer**.

Beispiel: Anna und Tom

Anna und Tom nutzen das Journal gemeinsam auf Annas Rechner. Anna hat ein Echtgeld-Konto (Kontonummer endet auf `1234`) und ein Paper-Konto (`5678`), Tom nur ein Paper-Konto (`9012`).

Text-Editor, Datei `users.toml` — beide Namen mit ihrem Präfix:

```
[anna]
prefix = "anna_"

[tom]
prefix = "tom_"
```



Text-Editor, Datei `accounts.toml` — jedes Konto einem Namen und einem Environment zuordnen:

```
[1234]
user = "anna"
environment = "money"

[5678]
user = "anna"
environment = "paper"

[9012]
user = "tom"
environment = "paper"
```

Text-Editor, Datei `.env` — die wichtigen Zeilen (alle anderen bleiben wie in der Vorlage):

```
DB_USER=root
DB_PASS=

DEFAULT_USER=anna

IMAP_USERNAME=anna@example.com
IMAP_PASSWORD=Passwort-von-Annas-Postfach
IMAP_SERVER=imap.gmx.com

IBKR_FLEX_QUERIES=PrüfcodeAnna:QueryIdAnnaMoney,PrüfcodeAnna:QueryIdAnnaPaper,PrüfcodeTom:QueryIdTomPaper
```

Hinweis

`TradingJournal-Weaver` liest **ein** Postfach. IBKR schickt Toms Berichte aber an die E-Mail-Adresse seines eigenen IBKR-Kontos — Tom richtet deshalb in seinem Mailprogramm eine **Weiterleitung** der IBKR-Mails an Annas Postfach ein (der Absender `donotreply@interactivebrokers.com` muss dabei erhalten bleiben). Über die Web-API ist das nicht nötig: Dort stehen einfach alle Prüfcode/Query-ID-Paare hintereinander.

Die Präfixe aus `users.toml` werden später auch für die Benutzerkonten der Weboberfläche gebraucht (Kapitel TradingJournal installieren) — dort muss für Anna `anna_` und für Tom `tom_` stehen.

Sicherheit: Die Dateien `.env` und `accounts.toml` enthalten Zugangsdaten bzw. Kontodaten und werden **niemals** mit hochgeladen (automatisch ausgeschlossen) — das ist so gewollt, nicht anfassen.



4. Datenbank-Tabellen anlegen

Dieser Befehl legt die eigentlichen Datenbank-Tabellen an, in denen später die Handelsdaten landen.

Wichtig: Läuft man ihn ein zweites Mal für dieselbe Kombination, werden bestehende Daten dort gelöscht und neu angelegt — beim allerersten Setup aber genau richtig.

Den eigenen Namen (aus Schritt 3, mit Unterstrich am Ende) einsetzen. **test immer anlegen** — die Weboberfläche startet in diesem Bereich, und er dient zum gefahrlosen Ausprobieren. Dazu die Environments für die eigenen IBKR-Konten, z. B. **money** für den Echtgeld-Bereich:

```
python -m tools.maintenance.reset_IBKR_DB test dein_name_  
python -m tools.maintenance.reset_IBKR_DB money dein_name_
```

Für weitere Environments (z.B. zusätzlich Paper-Trading) den Befehl mit dem jeweils anderen Environment-Namen wiederholen:

```
python -m tools.maintenance.reset_IBKR_DB paper dein_name_
```

Beispiel Anna und Tom — beide bekommen *test*, dazu Anna *money* und *paper*, Tom nur *paper*:

```
python -m tools.maintenance.reset_IBKR_DB test anna_  
python -m tools.maintenance.reset_IBKR_DB money anna_  
python -m tools.maintenance.reset_IBKR_DB paper anna_  
python -m tools.maintenance.reset_IBKR_DB test tom_  
python -m tools.maintenance.reset_IBKR_DB paper tom_
```

5. Automatisierung einrichten

TradingJournal-Weaver kann die Datenverarbeitung automatisch für mehrere Personen/Environments auf einmal erledigen — welche das auf **diesem** Rechner sind, steht in einer Datei namens **pipeline.toml**, die noch nicht existiert. Aus einer Vorlage anlegen:

```
copy pipeline.toml.example pipeline.toml
```

Jetzt **pipeline.toml** im Text-Editor öffnen und den Platzhalter-Abschnitt anpassen — eigenen Namen (ohne Unterstrich) und die in Schritt 4 angelegten Environments eintragen:



```
[users.dein_name]
environments = ["money"]

[[test]]
user = "dein_name"
```

Das Test-Environment steht **nicht** in der Liste `environments`, sondern in einem eigenen `[[test]]` - Block — die Vorlage enthält ihn schon, dort nur den Namen anpassen.

Beispiel Anna und Tom — die Environments aus Schritt 4, TEST je Person als eigener Block:

```
[users.anna]
environments = ["money", "paper"]

[users.tom]
environments = ["paper"]

[[test]]
user = "anna"

[[test]]
user = "tom"
```

Der `[[test]]` -Block schadet im Alltag nicht: TEST liest nur Dateien, die man selbst nach `C:\Trade_XML\test\<benutzer>\incoming\` legt (siehe Alltag mit dem TradingJournal, Abschnitt „Wenn etwas nicht stimmt“), und wird für den Testlauf in Schritt 6 gebraucht.

Wichtig: Hier nur Environments eintragen, für die Schritt 4 auch wirklich schon gelaufen ist — sonst bricht die automatische Verarbeitung später mit einer Fehlermeldung ab, weil die zugehörigen Tabellen fehlen.

6. Ersten Testlauf machen

Erst eine „Trockenübung“, die nichts wirklich ausführt, nur zeigt was passieren würde:

```
python run_pipeline.py --dry-run
```

Sieht das plausibel aus (die eigenen Environments aus Schritt 5 tauchen auf), ein echter Testlauf — den eigenen Namen einsetzen:



```
python run_pipeline.py --users dein_name --envs test
```

7. Praktisch: Schnellzugriff einrichten

Damit man beim nächsten Mal nicht wieder manuell in den Ordner wechseln und die Arbeitsumgebung (Schritt 2) aktivieren muss, gibt es eine Desktop-Verknüpfung, die beides in einem Doppelklick erledigt.

Einmalig einrichten — im PowerShell-Fenster, das noch im `TradingJournal-Weaver` -Ordner steht:

```
.\create-desktop-shortcut.ps1
```

Falls eine Fehlermeldung mit den Worten „Ausführung von Skripts“ oder „deaktiviert“ erscheint:

Windows blockiert auf frisch eingerichteten Rechnern zunächst das Ausführen von `.ps1` -Dateien.

Einmalig beheben — PowerShell **als Administrator** öffnen (Windows-Taste → `PowerShell` eintippen → Rechtsklick auf den Treffer → „**Als Administrator ausführen**“) und eintippen:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

Danach ein normales (nicht als Administrator geöffnetes) PowerShell-Fenster im `TradingJournal-Weaver` -Ordner öffnen und den Befehl von oben nochmal ausführen.

Danach liegt auf dem Desktop eine Verknüpfung namens „**TradingJournal-Weaver Pipeline**“. Doppelklick darauf öffnet PowerShell bereits im richtigen Ordner mit aktivierter Arbeitsumgebung — dort dann direkt z.B. `python run_pipeline.py` eintippen.

8. Alte Daten nachholen (einmalig)

Der tägliche Bericht enthält nur die letzten Tage (siehe IBKR einrichten — Flex Queries und Kontoauszug, Abschnitt 2). Alles, was **vor** der Installation gehandelt wurde, fehlt also zunächst im Journal — auch Positionen, die heute noch offen sind. Diese alten Daten holst du **einmal** von Hand bei IBKR und legst sie dem ersten echten Lauf bei.

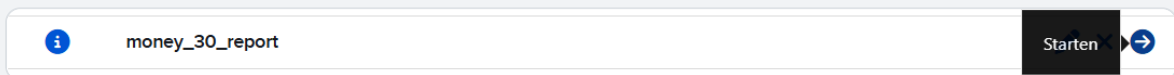


Wichtig

Alte Daten vor dem ersten echten Lauf einspielen — bzw. spätestens im selben Lauf wie die ersten täglichen Berichte. Innerhalb eines Laufs ordnet **TradingJournal-Weaver** alle Trades chronologisch, egal in welcher Datei sie stehen. Kommen alte Trades dagegen erst **nach** neueren Trades dazu, die schon verarbeitet sind, stimmen Positionen und Kampagnen nicht mehr (Beispiel und Abhilfe: Alltag mit dem TradingJournal, Abschnitt 1). Der Testlauf mit **--envs test** aus Schritt 6 ist unbedenklich, er berührt die echten Environments nicht.

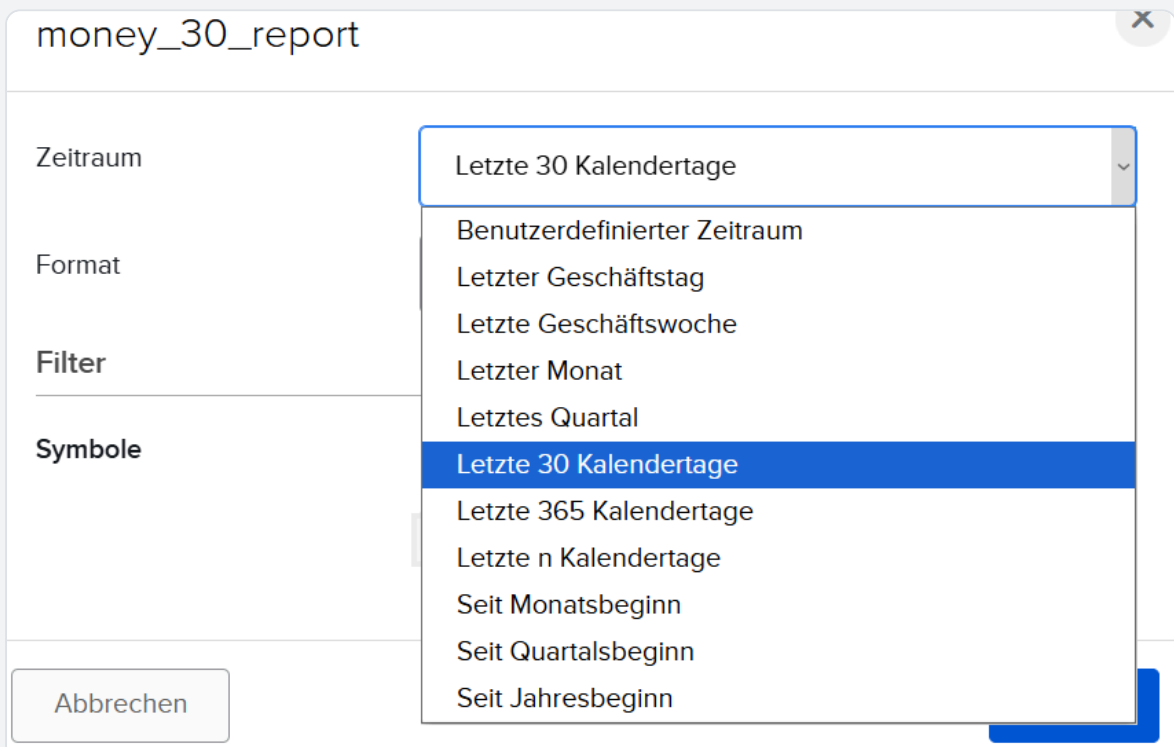
Schritt 1 — Bericht bei IBKR erzeugen (im Browser, IBKR-Kundenportal):

1. **Performance & Berichte** → Reiter **Flex Queries** öffnen.
2. Bei der Flex Query für die E-Mail (aus Kapitel „IBKR einrichten“, Abschnitt 2) ganz rechts auf den Pfeil **Starten** (→) klicken.



Flex Query starten

3. Im folgenden Fenster bei **Zeitraum** den Eintrag **Benutzerdefinierter Zeitraum** wählen. Für genau das letzte Jahr geht auch **Letzte 365 Kalendertage**.



Auswahl des Zeitraums



4. **Von (Datum)** und **Bis (Datum)** eintragen, **Format** auf **XML** lassen, den **Filter** leer lassen (sonst kommen nur einzelne Symbole) und auf **Starten** klicken. IBKR erstellt den Bericht, der Browser lädt ihn als `.xml` -Datei herunter.

money_30_report

Zeitraum

Benutzerdefinierter Zeitraum

Von (Datum)

2024-06-04

Bis (Datum)

2025-06-20

Der Datumsbereich darf nicht mehr als 365 Tage betragen.

Format

XML

Filter

Symbole

+ Symbol hinzufügen

☐ Derivate einbeziehen

Abbrechen

Starten

Benutzerdefinierter Zeitraum — hier mit zu langem Datumsbereich

Der Name der Query und der dort gespeicherte Zeitraum (z. B. 30 Tage) spielen dabei keine Rolle: Der Name dient nur der Übersicht, und beim händischen Starten gilt der Zeitraum, den du jetzt angibst. Die gespeicherte Query bleibt dabei unverändert.



Hinweis

Wie weit es zurückgeht:

- Je Abruf sind höchstens **365 Tage** möglich. Bei einem längeren Bereich erscheint *Der Datumsbereich darf nicht mehr als 365 Tage betragen* (siehe Bild). Für eine längere Vorgeschichte den Abruf Jahr für Jahr wiederholen, z. B. 2024-01-01 bis 2024-12-31, dann 2025-01-01 bis 2025-12-31 und so weiter. Mehrere Dateien sind kein Problem.
- **Paper-Konten:** IBKR liefert nur die **letzten rund 180 Tage** (etwa 6 Monate). Ältere Paper-Daten sind nicht mehr abrufbar — der Kalender lässt frühere Tage gar nicht erst auswählen (siehe Bild unten).
- **Echtgeld-Konten** reichen deutlich weiter zurück — IBKR stellt Kontodaten mehrere Jahre online bereit. Ein Datum vor der Kontoeröffnung ist sinnlos; am einfachsten beim Eröffnungsjahr beginnen.

paper_30_report ✕

Zeitraum

Benutzerdefinierter Zeitraum ▼

Von (Datum)

2026-10-19

Bis (Datum)

2026-10-19

Format

April 2026 »

Filter

Symbole

Su	Mo	Tu	We	Th	Fr	Sa
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Abbrechen

Starten

Paper-Konto: Am 20.10.2026 ist der 23.04.2026 der früheste wählbare Tag, alles davor ist ausgegraut

**Tipp**

Überlappungen sind kein Problem: Liegt das Bis-Datum mitten in einem Zeitraum, den auch der tägliche Bericht liefert, erkennt der Import die doppelten Trades und überspringt sie. Lieber großzügig bis heute abrufen als eine Lücke lassen.

Das wiederholst du für **jedes Konto** (Echtgeld, Paper, ...) — wie bei der Einrichtung im Kundenportal des jeweiligen Kontos.

Schritt 2 — Dateien ablegen (im Windows-Explorer):

Alle heruntergeladenen `.xml` -Dateien in den Ordner `C:\Trade_XML\incoming_mail` kopieren. Den Ordner legt `TradingJournal-Weaver` beim ersten Lauf selbst an; existiert er noch nicht, ihn einfach anlegen. Der Dateiname ist egal, er muss nur auf `.xml` enden: Zu welchem Benutzer und Environment eine Datei gehört, liest der Import aus der Kontonummer in der Datei. Die Kontonummer muss deshalb in `accounts.toml` stehen (Schritt 3), sonst wird die Datei übersprungen.

Schritt 3 — verarbeiten (PowerShell im `TradingJournal-Weaver` -Ordner mit aktivierter Arbeitsumgebung, z. B. über die Verknüpfung aus Schritt 7):

```
python run_pipeline.py
```

Das ist der normale Lauf: Er holt zusätzlich die aktuellen Berichte aus dem Postfach und verarbeitet alles zusammen in der richtigen Reihenfolge. Danach liegen die Dateien nicht mehr in `incoming_mail`, sondern in `C:\Trade_XML\<environment>\<benutzer>\processed` (bzw. `deleted`, wenn eine Datei keine neuen Trades enthielt), und die alten Trades sind im Journal sichtbar.

Und die Kontostände? Die Schritte oben holen nur die **Trades** nach (inkl. Verfall/Zuteilung von Optionen und Splits). Gewinn/Verlust, Kampagnen und Statistiken reichen damit so weit zurück wie die nachgeholten Trades. Die Kontostände (NAV, Cash, TWRR) kommen dagegen nur aus dem **täglichen** Kontoauszug (CSV) — und der lässt sich nur **Tag für Tag** nachholen:

- Im Kundenportal unter **Performance & Berichte** → **Kontoauszüge** beim eigenen, benutzerdefinierten Kontoauszug (aus Kapitel „IBKR einrichten“, Abschnitt 6) als Zeitraum **Täglich** wählen, ein **Datum** eintragen und **CSV herunterladen**.
- Die Datei nach `C:\Trade_CSV\incoming` kopieren — der Dateiname ist egal, die Zuordnung läuft auch hier über die Kontonummer in der Datei. Der nächste Lauf liest sie ein; die Reihenfolge der Tage spielt keine Rolle.
- Für jeden Tag ein eigener Download — für ein ganzes Jahr wären das rund 250. Sinnvoll ist das daher eher für ein paar Tage, z. B. um eine Lücke zu schließen. Beim **Paper-Konto** gilt dieselbe Grenze wie oben: höchstens die letzten rund 180 Tage.



Hinweis

Ein **Monats- oder Jahreskontoauszug** bzw. ein Kontoauszug mit *Benutzerdefiniertem Zeitraum* hilft nicht: Er fasst einen ganzen Zeitraum zusammen statt einen Tag. Der Import meldet *Kein Datum im CSV – übersprungen* und lässt die Datei in `incoming` liegen – dort einfach löschen. Wer auf die Kontostände der Vergangenheit verzichtet, verliert nur den Anfang von NAV-Verlauf, TWRR und Max Drawdown; der Kontoauszug ist ohnehin optional, siehe IBKR einrichten – Flex Queries und Kontoauszug, Abschnitt 6.

Zusammenfassung – künftig starten

Nach der Ersteinrichtung, für den täglichen Gebrauch:

1. Doppelklick auf die Desktop-Verknüpfung „TradingJournal-Weaver Pipeline“ (siehe Schritt 7)
2. Darin eintippen: `python run_pipeline.py`

Weiterführende Dokumentation

Datei	Inhalt
Technisches Handbuch: Installation: TradingJournal-Weaver	Dieselben Schritte, technischer/kompakter
<code>docs/handbuch/technik/30-befehlsreferenz.md</code> (im <code>TradingJournal-Weaver</code> -Ordner)	Vollständige Befehlsreferenz aller Scripts
TradingJournal installieren	Einrichtung der Weboberfläche (nächster Schritt)

TradingJournal installieren

Diese Anleitung richtet die Weboberfläche (TradingJournal) auf einem Rechner ein — Schritt für Schritt, mit Erklärungen, auch wenn du bisher wenig Erfahrung mit Terminal-Befehlen hast. Es gibt dazu auch eine kürzere, technischere Version: Technisches Handbuch: Installation: TradingJournal. Beide führen zum selben Ergebnis.

Bevor du hier startest: Die Grundausrüstung (XAMPP, Composer, Node.js, ...) muss bereits installiert sein — siehe Windows vorbereiten. Und die Datenimport-Pipeline sollte schon eingerichtet sein — siehe TradingJournal-Weaver installieren.

Inhalt

1. Projekt herunterladen
2. Konfigurationsdatei anlegen
3. Datenbank-Tabellen anlegen
4. Ersten Benutzer anlegen
5. Optik der Seite bauen
6. Starten und öffnen
7. Praktisch: Schnellzugriff einrichten

1. Projekt herunterladen

Ein PowerShell-Fenster öffnen (Windows-Taste drücken, `PowerShell` eintippen, Enter) und in den Ordner wechseln, in dem auch `TradingJournal-Weaver` schon liegt (z.B. `C:\Users\<dein Name>\trading`):

```
cd C:\Users\<dein Name>\trading
```

Falls das Projekt dort noch nicht liegt, herunterladen („klonen“):

```
git clone https://github.com/mike-1140/TradingJournal.git
cd TradingJournal
```

Jetzt zwei Befehle, die automatisch alle benötigten Zusatz-Bausteine herunterladen — das kann je nach Internetverbindung ein bis zwei Minuten dauern:



```
composer install  
npm install
```

Falls bei `composer install` eine Fehlermeldung mit dem Wort „zip“ erscheint: Das bedeutet, eine PHP-Einstellung namens `zip` ist noch nicht aktiviert. Wie man das behebt, steht in Windows vorbereiten im Abschnitt „Eine PHP-Einstellung anpassen“. Danach diesen Schritt hier einfach nochmal ausführen.

2. Konfigurationsdatei anlegen

Jedes Projekt braucht eine kleine Datei namens `.env`, in der z.B. die Datenbank-Zugangsdaten stehen. Diese Datei liegt **nicht** im Internet (aus Sicherheitsgründen), sondern wird lokal aus einer Vorlage erstellt.

Wichtig

Das ist **eine eigene `.env`-Datei**, nicht dieselbe wie bei `TradingJournal-Weaver`. Beide Programme haben jeweils ihre eigene `.env` im eigenen Ordner, mit unterschiedlichem Inhalt und teils unterschiedlichen Namen für dieselbe Sache (Weaver: `DB_USER` / `DB_PASS`, hier: `DB_USERNAME` / `DB_PASSWORD`). Also die Vorlage **im Ordner `TradingJournal`** kopieren und die Datei aus dem Weaver nicht hierher kopieren.

PowerShell, im Ordner `TradingJournal`:

```
copy .env.example .env  
php artisan key:generate
```

Die Vorlage ist bereits für eine frische XAMPP-Installation eingestellt (Benutzer `root`, kein Passwort) — **es muss nichts geändert werden**. Nur wer für MySQL ein Passwort vergeben hat, öffnet die Datei `.env` im Text-Editor (Rechtsklick → **Öffnen mit** → **Editor**) und trägt es bei `DB_PASSWORD=` und `TRADING_DB_PASSWORD=` ein.

3. Datenbank-Tabellen anlegen

Dieser Befehl legt die Tabellen an, die die Weboberfläche selbst braucht (z.B. für die Benutzer):



```
php artisan migrate
```

Wichtig: Ohne diesen Schritt gibt es keine Benutzer, und die Seite startet nicht.

Die eigentlichen Handelsdaten-Tabellen kommen von einem anderen Programm ([TradingJournal-Weaver](#)) — die sollten laut der [TradingJournal-Weaver](#) -Anleitung schon angelegt worden sein. Falls nicht, jetzt nachholen (aus dem [TradingJournal-Weaver](#) -Ordner heraus, `<dein Name>_` durch den eigenen gewählten Namen ersetzen):

```
cd ..\TradingJournal-Weaver
python -m tools.maintenance.reset_IBKR_DB test <dein Name>_
python -m tools.maintenance.reset_IBKR_DB money <dein Name>_
cd ..\TradingJournal
```

4. Ersten Benutzer anlegen

Zwei Befehle, nacheinander. Der erste bereitet eine kleine technische Sache vor (nur einmal nötig, auch bei mehreren Benutzern):

```
php artisan setup:auth
```

(Erscheint dabei „[User ID 1 nicht gefunden](#)“ — das ist an dieser Stelle normal und kein Fehler, weil noch kein Benutzer existiert.)

Jetzt den ersten Benutzer anlegen — Name, E-Mail und Passwort durch eigene Werte ersetzen, `<dein Name>_` durch denselben Namen wie in Schritt 3:

```
php artisan auth:create-user "Dein Name" deine@mail-adresse.at "EinSicheresPasswort" --
prefix=<dein Name>_ --admin
```

Das Wort `--admin` am Ende macht diesen ersten Benutzer zum Administrator. Auf dem eigenen PC wird er beim Öffnen der App automatisch angemeldet und kann über einen Dropdown zu weiteren Benutzern wechseln (siehe Erster Start — UI-Erstkonfiguration, Abschnitt 3). Deshalb sollte der eigene Zugang als **erster** angelegt werden.



5. Optik der Seite bauen

Dieser Befehl baut Farben, Icons und Layout der Weboberfläche zusammen:

```
npm run build
```

6. Starten und öffnen

Die einfachste Variante, um die Seite auszuprobieren:

```
php artisan serve
```

Danach im Browser <http://localhost:8000> öffnen. Ein Login ist auf dem eigenen PC nicht nötig: Die App meldet automatisch den in Schritt 4 angelegten Benutzer an.

Zu beachten: Diese Variante läuft nur, solange das PowerShell-Fenster offen bleibt. Fenster geschlossen → Seite nicht mehr erreichbar, dann `php artisan serve` erneut eintippen.

Es gibt eine fortgeschrittenere Variante, bei der die Seite dauerhaft im Hintergrund läuft (über Apache aus dem XAMPP-Paket, mit einer eigenen Adresse wie <http://trading.local>) — das lohnt sich, wenn man die Seite oft nutzt und nicht jedes Mal ein Terminal offen haben will. Diese Einrichtung ist etwas technischer (Konfigurationsdatei bearbeiten) — Details dazu in der technischen Version Technisches Handbuch: Installation: TradingJournal, Abschnitt „7. Starten, Variante B“. Am besten mit etwas Erfahrung oder Unterstützung einrichten, für den Anfang reicht `php artisan serve` völlig aus.

7. Praktisch: Schnellzugriff einrichten

Damit man nicht jedes Mal manuell in den richtigen Ordner navigieren muss, gibt es ein kleines Hilfsmittel: eine Verknüpfung auf dem Desktop, die per Doppelklick direkt ein PowerShell-Fenster im richtigen Ordner öffnet.

Einmalig einrichten — im PowerShell-Fenster, das noch im `TradingJournal` -Ordner steht:

```
.. \create-desktop-shortcut.ps1
```

Falls eine Fehlermeldung mit den Worten „Ausführung von Skripts“ oder „deaktiviert“ erscheint: Windows blockiert auf frisch eingerichteten Rechnern zunächst das Ausführen von `.ps1` -Dateien. Einmalig beheben — PowerShell **als Administrator** öffnen (Windows-Taste → `PowerShell` eintippen →



Rechtsklick auf den Treffer → „**Als Administrator ausführen**“) und eintippen:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

Danach ein normales (nicht als Administrator geöffnetes) PowerShell-Fenster im `TradingJournal` - Ordner öffnen und den Befehl von oben nochmal ausführen.

Danach liegt auf dem Desktop eine Verknüpfung namens „**TradingJournal**“. Doppelklick darauf öffnet PowerShell direkt im richtigen Ordner — dort dann z.B. `php artisan serve` eintippen.

Zusammenfassung — künftig starten

Nach der Ersteinrichtung, für den täglichen Gebrauch:

1. Doppelklick auf die Desktop-Verknüpfung „TradingJournal“ (siehe Schritt 7)
2. Darin eintippen: `php artisan serve`
3. Im Browser `http://localhost:8000` öffnen

Weiterführende Dokumentation

Datei	Inhalt
Technisches Handbuch: Installation: TradingJournal	Dieselben Schritte, technischer/kompakter, plus fortgeschrittene Themen (Apache-VirtualHost)
Erster Start — UI-Erstkonfiguration	Was nach dem ersten Start einzustellen ist
Installation: TradingJournal-Weaver	Setup der Datenimport-Pipeline

Erster Start — UI-Erstkonfiguration

Was nach der technischen Installation (siehe Technisches Handbuch: Installation: TradingJournal bzw. TradingJournal installieren) einmalig einzustellen ist, bevor die App produktiv genutzt wird — und was die einzelnen Bedienelemente in der Oberfläche eigentlich bedeuten.

TradingJournal verhält sich unterschiedlich, je nachdem, wo es läuft. Maßgeblich ist die Einstellung `APP_ENV` in der `.env` von TradingJournal:

	Auf dem eigenen PC (<code>APP_ENV=local</code> , so wie bei der Installation in „TradingJournal installieren“)	Auf einem Server (<code>APP_ENV=production</code> , siehe Technisches Handbuch „Server-Betrieb“)
Anmelden	kein Login — die App meldet automatisch den zuerst angelegten Benutzer an	Login-Seite, jede Person mit eigenem Zugang
Andere Benutzer ansehen	als Admin über den Benutzer-Dropdown (Abschnitt 3)	nicht möglich

1. Anmelden

Benutzer werden beim Setup per `php artisan auth:create-user` angelegt („Technisches Handbuch: Installation: TradingJournal“ Schritt 5 bzw. „TradingJournal installieren“ Schritt 4). Es gibt bewusst **keine Registrierungsseite** — ein neuer Zugang entsteht immer nur über diesen einen Befehl auf der Kommandozeile, nie über die Weboberfläche selbst. Das ist Absicht: Es soll niemand „einfach so“ ein Konto anlegen können.

Auf dem eigenen PC erscheint keine Login-Seite: Beim Öffnen der App wird automatisch der **zuerst angelegte Benutzer** angemeldet (der mit der internen Nummer 1). Deshalb sollte das der eigene Admin-Zugang sein — so wie in der Installationsanleitung mit `--admin` angelegt.

Auf einem Server meldet sich jede Person mit E-Mail-Adresse und Passwort an, die beim Anlegen vergeben wurden.

2. Der Environment-Switcher (oben in der Navigation)

Was ist ein „Environment“?

Ein Environment ist ein abgetrennter Datenbereich — z.B. `MONEY` für echtes Geld, `PAPER` für simuliertes Übungs-Trading, `TEST` zum Ausprobieren. Jedes Environment hat seine eigenen, komplett getrennten Daten; ein Trade im TEST-Bereich taucht niemals im MONEY-Bereich auf und umgekehrt. Der Dropdown oben in der Navigation schaltet zwischen diesen Bereichen um.



Warum sieht man dort nicht automatisch alle Environments?

Der Dropdown zeigt nur die Environments an, für die beim eigenen Nutzer-Konto auch wirklich schon Datenbank-Tabellen angelegt wurden (Schritt „Datenbank-Tabellen anlegen“ beim `TradingJournal-Weaver` -Setup). Fehlt ein Environment im Dropdown, ist das **kein Fehler** — es bedeutet nur, dass für dieses Environment noch keine Tabellen existieren. Sobald das nachgeholt wird (`reset_IBKR_DB.py` `<env>` `<prefix>` in `TradingJournal-Weaver`), erscheint es automatisch.

TEST steht immer im Dropdown — als sicherer Ausgangspunkt zum Ausprobieren. Damit es auch funktioniert, braucht es wie jedes Environment Tabellen; die Installationsanleitung legt sie deshalb immer mit an (TradingJournal-Weaver installieren, Schritt 4). Fehlen sie doch, zeigt die App beim Auswählen von TEST eine Fehlermeldung.

Falle, die für Verwirrung sorgen kann

Nach dem nachträglichen Anlegen eines Environments kann es bis zu **10 Minuten** dauern, bis es im Dropdown erscheint (die App merkt sich das Ergebnis kurzzeitig, um nicht bei jedem Klick erneut in der Datenbank nachzusehen). Wer es sofort sehen will, statt zu warten: auf der Kommandozeile im `TradingJournal` -Ordner `php artisan cache:clear` ausführen.

Startwert: Beim Öffnen der App (auf dem Server: nach jedem Login) steht der Schalter zunächst auf **TEST**. Wer lieber direkt in einem anderen Bereich landen möchte, trägt in der `.env` von TradingJournal z. B. `TRADING_DEFAULT_ENV=MONEY` ein. Hat der Benutzer im Startwert keine Tabellen, startet die App stattdessen im ersten Environment, für das es welche gibt (Reihenfolge MONEY, PAPER, TEST, AUTOM_MONEY, AUTOM_PAPER).

3. Mehrere Personen nutzen dasselbe TradingJournal

Dieser Punkt ist nur relevant, wenn **mehr als eine Person** dieselbe Installation nutzt (z.B. zwei Familienmitglieder mit je eigenem IBKR-Konto). Bei nur einer Person kann dieser Abschnitt übersprungen werden.

Auf einem Server ist das einfach: Jede Person hat ihren eigenen Zugang, meldet sich damit an und sieht nur ihre eigenen Daten. Einen Wechsel zwischen Benutzern gibt es dort nicht.

Auf dem eigenen PC gibt es keine Anmeldung (Abschnitt 1) — wie kommt die zweite Person an ihre Daten? Über den **Benutzer-Dropdown** auf dem Dashboard, rechts neben dem Zahnrad. Er erscheint nur beim Admin-Zugang (angelegt mit `--admin`):

1. Beim Öffnen der App ist automatisch der Admin angemeldet, der Environment-Schalter steht auf TEST. Der Admin sieht seine eigenen Environments; im Benutzer-Dropdown stehen zusätzlich die anderen Benutzer zur Auswahl.
2. Ein Klick auf einen anderen Benutzer wechselt zu dessen Daten: Der Environment-Schalter zeigt jetzt **dessen** Environments, Dashboard und alle Listen zeigen dessen Trades. Ein Passwort braucht



es dafür nicht.

3. Als dieser Benutzer gibt es keinen Benutzer-Dropdown mehr — weitere Benutzer lassen sich von hier aus nicht auswählen.
4. **Zurück zum Admin:** rechts oben auf **Abmelden** klicken (Symbol Pfeil aus Kasten, neben dem Benutzernamen). Auf dem eigenen PC führt das nicht zu einer Login-Seite — die App meldet sofort wieder automatisch den Admin an.

Wer gerade angemeldet ist, steht immer rechts oben neben dem Abmelden-Knopf.

Für einen Zwei-Personen-Haushalt reicht es also, wenn **eine** Person `--admin` bekommt — die zweite Person braucht das nicht. Ihr Zugang wird trotzdem mit `auth:create-user` angelegt (mit eigenem Präfix), damit es ihre Daten überhaupt gibt.

4. Dashboard-Kacheln anpassen

Über das Zahnrad-Symbol auf dem Dashboard lässt sich auswählen, welche Info-Kacheln (z.B. Profit Factor, größter Gewinn/Verlust, ...) angezeigt werden. Praktisch, um das Dashboard auf das zu reduzieren, was einen wirklich interessiert.

Die Auswahl wird **je Environment** und je Benutzer gespeichert — im Fenster steht „Kacheln für MONEY“, „Kacheln für PAPER“ usw. So kann z. B. das Paper-Dashboard schlanker sein als das Echtgeld-Dashboard. Solange nichts ausgewählt wurde, sind alle Kacheln eingeschaltet. Kontostands-Kacheln (NAV, Cash, TWRR) erscheinen nur, wenn für das Environment ein Kontoauszug eingelesen wurde (siehe IBKR einrichten — Flex Queries und Kontoauszug, Abschnitt 6).

Alltag mit dem TradingJournal

Was im laufenden Betrieb zu tun ist: neue Daten holen und verarbeiten, sie in der App durchsehen und ergänzen — und was zu tun ist, wenn etwas nicht stimmt. Wie die einzelnen Seiten der App aussehen, zeigt Rundgang durch die App; was im Hintergrund genau passiert, das technische Handbuch (Architektur und Import-Pipeline).

1. Daten holen und verarbeiten

So startest du einen Lauf — Doppelklick auf die Desktop-Verknüpfung „**TradingJournal-Weaver Pipeline**“ (aus TradingJournal-Weaver installieren, Schritt 7). Im PowerShell-Fenster, das sich öffnet:

```
python run_pipeline.py
```

Danach im Browser die App neu laden — die neuen Daten sind da.

Was dabei passiert — der Lauf arbeitet sechs Schritte nacheinander ab:

- 1 **Mails abholen:** neue IBKR-Berichte (Flex Query als XML, Kontoauszug als CSV) aus dem Postfach holen; die Mails werden danach gelöscht.
- 2 **Web-API abfragen:** dieselben Trades auf dem zweiten Weg holen — nur, wenn der Abruf per Web-API eingerichtet ist.
- 3 **Kontostände einlesen:** die Kontoauszüge auswerten (Nettovermögen, Cash, Tagesrendite).
- 4 **Trades einlesen:** alle neuen Berichte lesen, jedem Benutzer und Environment zuordnen und doppelt gelieferte Trades überspringen.
- 5 **Trades verarbeiten:** offene und geschlossene Positionen bilden, Gewinn/Verlust berechnen, Kampagnen fortschreiben.
- 6 **Sammeldatei ergänzen:** die eingelesenen Berichte zusätzlich in einer Sammeldatei je Environment ablegen — daraus lässt sich später alles neu aufbauen (Abschnitt 4).



Welche Benutzer und Environments auf diesem Rechner verarbeitet werden, steht in der `pipeline.toml` (TradingJournal-Weaver installieren, Schritt 5). Die einzelnen Schritte mit ihren technischen Namen und Optionen beschreibt die Befehlsreferenz. Wer den Lauf automatisch starten lassen möchte (z. B. täglich per Windows-Aufgabenplanung), findet die Anleitung in Eigener Server, Abschnitt 4 — sie gilt genauso für den eigenen PC.

Muss das täglich laufen? — Nein, auch einmal im Monat genügt

Die Pipeline muss nicht jeden Tag laufen. Wer nur einmal pro Woche oder einmal im Monat nachsehen möchte, startet sie einfach dann — es geht nichts verloren:

- **Die Mails warten im Postfach.** IBKR schickt die Berichte weiter jeden Tag, sie bleiben im Posteingang liegen, bis der nächste Lauf sie abholt — dann alle auf einmal, Flex Queries wie Kontoauszüge. Jeder Tag hat seinen eigenen Kontoauszug, die Kontostände sind also lückenlos. Gelesen werden dürfen die Mails, gelöscht oder in einen anderen Ordner verschoben nicht (auch nicht durch eine Regel im Mailprogramm); und das Postfach darf nicht volllaufen.
- **Das Ergebnis ist dasselbe wie bei täglichen Läufen.** Innerhalb eines Laufs werden alle Trades chronologisch verarbeitet, egal aus welcher Mail sie stammen. Auch die Kampagnen-Regel „10 Tage ohne Aktivität → CLOSED“ richtet sich nach den Daten der Trades, nicht nach dem Tag des Laufs.
- **Der Abruf per Web-API hilft dabei nicht.** Er holt nur den Zeitraum, der in der API-Query eingestellt ist (z. B. die letzten 8 Tage), gerechnet ab dem Moment des Abrufs. Bei monatlichen Läufen liefert die tägliche Mail die Daten; die Web-API ist nur für Ungeduldige, die morgens schon die Vortagesdaten sehen wollen.

Der einzige Nachteil: Das Journal zeigt immer den Stand des letzten Laufs. Dashboard, Kampagnen und die Liste der noch nicht dokumentierten Trades im Trade Log sind entsprechend alt.

Tipp

Wer gar keine tägliche Mail möchte, kann die Flex Query auch einmal im Monat von Hand im Kundenportal starten (Zeitraum **Letzter Monat** oder ein benutzerdefinierter Zeitraum) und die Datei nach `C:\Trade_XML\incoming_mail` legen — genau wie beim Nachholen alter Daten in TradingJournal-Weaver installieren, Abschnitt 8. Die Kontostände fehlen dann allerdings, denn der Kontoauszug kommt nur täglich per Mail.

Spielt die Reihenfolge der Dateien eine Rolle?

Innerhalb eines Laufs nicht. Der Import legt alle Trades aus allen Dateien zuerst in einem Zwischenspeicher ab. Die Verarbeitung nimmt sie danach streng nach Handelszeitpunkt, egal aus welcher Datei, welchem Abrufweg (Mail oder Web-API) und in welcher Reihenfolge die Dateien ankamen. Auch doppelt gelieferte Trades sind kein Problem — sie werden beim Import erkannt und übersprungen.

Über mehrere Läufe hinweg schon: Was einmal verarbeitet ist, wird nicht neu sortiert. Kommen Trades erst an, **nachdem** jüngere Trades schon verarbeitet wurden, werden sie trotzdem gegen den aktuellen Stand gebucht. Ein Beispiel:



Handelstag	Trade	Verarbeitet
02.03.	Kauf 100 Aktien	erst im zweiten Lauf (alte Daten, nachgeholt)
10.03.	Verkauf 100 Aktien	im ersten Lauf

Im ersten Lauf findet der Verkauf keinen Bestand und wird als neue **Short-Position** eröffnet. Der nachträglich verarbeitete Kauf schließt diese Short-Position dann wieder. Am Ende ist der Bestand zwar null, aber Richtung, Daten und Kampagne stimmen nicht — aus einer Long-Position vom 02.03. bis 10.03. ist eine Short-Position geworden, eröffnet am 10.03. und geschlossen am 02.03.

Im normalen Betrieb passiert das nicht: Jeder Bericht enthält die letzten Tage (siehe IBKR einrichten — Flex Queries und Kontoauszug, Abschnitt 2), ein Trade ist also im ersten Bericht nach dem Handelstag enthalten — vor allen jüngeren Trades. Die Kontostände aus dem Kontoauszug sind ohnehin unabhängig voneinander, dort spielt die Reihenfolge nie eine Rolle.

Aufpassen muss man nur, wenn **alte Daten nachgeholt** werden: Sie gehören vor den ersten echten Lauf oder in denselben Lauf (siehe TradingJournal-Weaver installieren, Abschnitt 8). Dasselbe gilt sinngemäß, wenn Dateien länger liegen geblieben sind, z. B. weil ein Konto erst später in `accounts.toml` eingetragen wurde, während für dasselbe Environment schon neuere Trades verarbeitet waren.

Das Journal passt darauf auf: Liegt ein neu importierter Trade zeitlich vor dem zuletzt verarbeiteten, verarbeitet die Pipeline dieses Environment **nicht** und meldet im Schritt `process` einen Fehler mit Datum und Symbol des ältesten betroffenen Trades. Die anderen Environments laufen normal weiter. Ausgenommen sind Verfall, Zuteilung und Barausgleich von Optionen — IBKR liefert sie gelegentlich erst mit dem Bericht des Folgetags; das ist unkritisch und steht nur als Hinweis im Protokoll. Wer weiß, was er tut, kann die Verarbeitung erzwingen ([**PowerShell**] im Ordner `TradingJournal-Weaver`):

```
python -m src.process_ibkr_trades money dein_user_ --reihenfolge-ignorieren
```

Achtung

Ist es schon passiert, hilft nur ein **Neuaufbau des Environments** (`reset_IBKR_DB.py` und Import aus der Sammeldatei, siehe Abschnitt 4, „Alles neu aufbauen“). Die Sammeldatei ist nach Datum sortiert und enthält auch die nachgeholten Tage, beim Neuaufbau stimmt die Reihenfolge also wieder. Der Neuaufbau legt die Tabellen dieses Environments neu an — neben Trades und Kampagnen gehen dabei auch die Einträge im **Trade Log** und die **Kontostände** verloren. Die Kontostände lassen sich wiederherstellen, indem die verarbeiteten CSV-Dateien aus `C:\Trade_CSV\incoming\<benutzer>\processed` zurück nach `C:\Trade_CSV\incoming` kopiert werden; Trade-Log-Einträge vorher sichern (Abschnitt 4, „Datenbank sichern und wiederherstellen“).



So sieht ein Abruf aus

Die beiden Abrufe laufen normalerweise als Teil von `python run_pipeline.py` (Schritte `fetch_emails` und `fetch_api`), lassen sich aber auch einzeln starten. Die folgenden Protokolle stammen aus echten Läufen mit vier Konten (Echtgeld, Paper und je ein Konto für automatisierten Handel). Kontonummern, Query-IDs, Prüfcode-Endungen, Referenznummern und Mailadresse sind ersetzt, lange Zeilen gekürzt (...).

Abruf per Web-API

PowerShell (im `TradingJournal-Weaver` -Ordner, Arbeitsumgebung aktiv):



```
(.venv) PS C:\Users\<Benutzer>\Trading\TradingJournal-Weaver> python.exe -m
src.get_flexquery_api

=====
---- FlexQueryAPI (2026-10-20 - 07:14:59)
=====
4 Queries konfiguriert | Tokens: ['1111', '2222', '3333', '4444']
[07:14:59] Query 1000001: Anfrage OK nach 1 Versuch(en)/0.8s – HTTP 200,
ReferenceCode=1234567890, GetStatement-URL=...
[07:15:00] Query 1000001: Statement erhalten nach 1 Versuch(en)/0.2s – HTTP 200, 1,557
Zeichen
Query 1000001: Erfolgreich nach 1.0s gesamt →
C:\Trade_XML\incoming_api\U1234567_20261012_act_money_api_8_0001.xml
Token-Wechsel – warte 30s zur Sicherheit ...
[07:15:30] Query 1000002: Anfrage OK nach 1 Versuch(en)/0.3s – HTTP 200,
ReferenceCode=1234567891, GetStatement-URL=...
[07:15:31] Query 1000002: Noch in Verarbeitung/Bereitstellung [1019] (Versuch 1/5, seit
0.3s), warte 10s ...
[07:15:41] Query 1000002: Statement erhalten nach 2 Versuch(en)/10.5s – HTTP 200, 5,035
Zeichen
Query 1000002: Erfolgreich nach 10.7s gesamt →
C:\Trade_XML\incoming_api\DU1234567_20261012_act_paper_api_8_0002.xml
Token-Wechsel – warte 30s zur Sicherheit ...
[07:16:11] Query 1000003: Anfrage OK nach 1 Versuch(en)/0.2s – HTTP 200,
ReferenceCode=1234567892, GetStatement-URL=...
[07:16:11] Query 1000003: Statement erhalten nach 1 Versuch(en)/0.3s – HTTP 200, 1,563
Zeichen
Query 1000003: Erfolgreich nach 0.5s gesamt →
C:\Trade_XML\incoming_api\U7654321_20261012_act_autom_money_api_8_0003.xml
Token-Wechsel – warte 30s zur Sicherheit ...
[07:16:42] Query 1000004: Anfrage OK nach 1 Versuch(en)/0.2s – HTTP 200,
ReferenceCode=1234567893, GetStatement-URL=...
[07:16:42] Query 1000004: Statement erhalten nach 1 Versuch(en)/0.2s – HTTP 200, 1,563
Zeichen
Query 1000004: Erfolgreich nach 0.4s gesamt →
C:\Trade_XML\incoming_api\DU7654321_20261012_act_autom_paper_api_8_0004.xml
Fertig: 4 neue FlexQuery-Datei(en) gespeichert – Laufzeit gesamt 102.8s
Fertig, Dateien gespeichert: 4
```

So ist das zu lesen:

- **Tokens:** Vom Prüfcode werden nur die letzten vier Zeichen angezeigt. Vier verschiedene Prüfcodes bedeuten vier Konten mit je eigenem Kundenportal-Login.
- **Zwei Schritte je Query:** Zuerst die *Anfrage* (IBKR beginnt den Bericht zu erstellen und gibt eine Referenznummer zurück), dann das Abholen des fertigen *Statements*.



- **Zwei Versuche beim Paper-Konto sind normal.** Meldung `[1019]` heißt: IBKR stellt den Bericht noch bereit. Der Abruf wartet 10 Sekunden und versucht es erneut, höchstens fünfmal. Paper-Konten werden bei IBKR offenbar nachrangig bedient und brauchen öfter einen zweiten Anlauf.
- **„Token-Wechsel — warte 30s“:** Vor jedem Wechsel auf einen anderen Prüfcode wartet der Abruf absichtlich 30 Sekunden. Deshalb dauert der ganze Lauf rund 100 Sekunden, obwohl jede einzelne Query in ein bis elf Sekunden fertig ist.
- **Dateiname:** Kontonummer, Datum, Name der Query, letzte vier Ziffern der Query-ID. Das Datum ist der **erste** Tag im Bericht — hier der 12.10., weil die Query die letzten 8 Tage abrufen. Der Teil `_8_` stammt aus dem Namen der Query (`act_money_api_8`) und hat sonst keine Bedeutung.

Achtung

Die Web-API höchstens einmal am Tag abrufen. IBKR geht streng gegen vermeintliche Bots vor: Zu viele Anfragen kurz hintereinander — besonders fehlgeschlagene — führen zu einer vorübergehenden Sperre (Fehler 1025, siehe Fehlerbehebung und häufige Fragen).

`TradingJournal-Weaver` schützt sich selbst: Ein zweiter Aufruf am selben Tag fragt IBKR nicht erneut, sondern meldet nur `Heute bereits geholt, übersprungen`. Trotzdem nicht mit wiederholten Aufrufen herumprobieren — neue Daten gibt es ohnehin erst nach dem nächsten Handelstag.

Abruf per E-Mail

PowerShell (im `TradingJournal-Weaver` -Ordner, Arbeitsumgebung aktiv) — hier nach drei Tagen ohne Lauf, gekürzt auf den ersten Tag:



```
(.venv) PS C:\Users\<Benutzer>\Trading\TradingJournal-Weaver> python.exe -m
src.get_flexquery_emails
=====
---- FlexQueryEmails (2026-10-20 - 07:31:49)
=====
Mail-Account: trading-mails@example.com
Anhang gespeichert: flex.1000011.paper_30_report.20261007.20261014.xml →
C:\Trade_XML\incoming_mail\flex.1000011.paper_30_report.20261007.20261014.xml
Anhang gespeichert: flex.1000012.money_30_report.20260915.20261014.xml →
C:\Trade_XML\incoming_mail\flex.1000012.money_30_report.20260915.20261014.xml
Anhang gespeichert: flex.1000013.autom_paper_8.20261007.20261014.xml →
C:\Trade_XML\incoming_mail\flex.1000013.autom_paper_8.20261007.20261014.xml
Anhang gespeichert: flex.1000014.autom_money_8.20261007.20261014.xml →
C:\Trade_XML\incoming_mail\flex.1000014.autom_money_8.20261007.20261014.xml
... (dasselbe für den 15. und 16.10.)
Anhang gespeichert: money_act.2000001.20261014.csv →
C:\Trade_CSV\incoming\money_act.2000001.20261014.csv
Activity CSV gespeichert: money_act.2000001.20261014.csv → incoming/
Anhang gespeichert: autom_money_act.2000002.20261014.csv →
C:\Trade_CSV\incoming\autom_money_act.2000002.20261014.csv
Activity CSV gespeichert: autom_money_act.2000002.20261014.csv → incoming/
Bereits vorhanden, uebersprungen: autom_money_act.2000002.20261014.csv
Anhang gespeichert: paper_act.2000003.20261014.csv →
C:\Trade_CSV\incoming\paper_act.2000003.20261014.csv
Activity CSV gespeichert: paper_act.2000003.20261014.csv → incoming/
... (dasselbe für den 15. und 16.10.)
Fertig: 12 FlexQuery-Datei(en), 9 Activity-CSV(s) gespeichert
Fertig, Dateien verarbeitet: 21
```

So ist das zu lesen:

- **Alles auf einmal:** Seit dem letzten Lauf sind drei Tage vergangen, also holt der Abruf die Berichte zu drei Handelstagen (14.–16.10.): 4 Flex Queries × 3 Tage = 12 XML-Dateien, dazu 3 Kontoauszüge × 3 Tage = 9 CSV-Dateien. Die abgeholten Mails werden aus dem Posteingang gelöscht.
- **Flex-Query-Dateien** (`.xml`) landen in `C:\Trade_XML\incoming_mail` , **Kontoauszüge** (`.csv`) in `C:\Trade_CSV\incoming` .
- **Dateiname der XML:** `flex.<Query-ID>.<Name der Query>.<von>.<bis>.xml` . Der Name sagt nichts über den Inhalt: `paper_30_report` enthält hier nur sieben Tage (07.–14.10.) — maßgeblich sind die beiden Datumsangaben, nicht die Zahl im Namen.
- **„Bereits vorhanden, uebersprungen“:** Derselbe Kontoauszug kam ein zweites Mal an und wird nicht doppelt gespeichert — harmlos.
- **Nur drei Kontoauszüge für vier Konten:** Für das vierte Konto ist kein Kontoauszug eingerichtet. Das ist erlaubt, er ist optional (siehe IBKR einrichten — Flex Queries und Kontoauszug, Abschnitt 6).



Delta und IV-Rang automatisch (optional)

Delta, implizite Volatilität (IV), IV-Rang (IVR) und IV-Perzentil (IVP) liefert IBKR in den Berichten nicht mit. Der Weaver kann sie **nachträglich** aus der Trader Workstation (TWS) berechnen — aus dem Kurs des Basiswerts in der Minute deiner Ausführung und dem Verlauf der Volatilität. Beim Trade selbst musst du dafür nichts tun.

Voraussetzungen, einmalig:

1. **[Trader Workstation]** *Datei* → *Globale Konfiguration* → *API* → *Einstellungen*: „ActiveX- und Socket-Clients aktivieren“ einschalten. Der Weaver verbindet sich nur lesend.
2. **[Text-Editor]** In der `pipeline.toml` den Schritt `enrich` nach `process` eintragen, z. B. `steps = ["fetch_emails", "fetch_api", "snapshot_csv", "import", "process", "enrich", "summary"]`.

Danach rechnet jeder Pipeline-Lauf die Werte für neue Options-Trades aus, sofern die TWS gerade läuft und angemeldet ist. Sie erscheinen im **Log** und auf der Kampagnen-Seite: Delta je Leg sowie Kurs und IVR/IVP bei Eröffnung und Schließung (Hinweis „aus IBKR berechnet“). Läuft sie nicht, wird der Schritt übersprungen und beim nächsten Mal nachgeholt. Ältere Trades lassen sich einmalig nachholen (**[PowerShell]** im Ordner `TradingJournal-Weaver`):

```
python -m src.enrich_greeks --seit 2025-01-01
```

2. Nach dem Import: ein typischer Durchgang

Die Daten sind nur die eine Hälfte eines Journals — die andere ist, was du dazu festhältst. Ein Durchgang nach dem Import, z. B. am Morgen mit den Daten vom Vortag:

- 1 **Dashboard** — wie steht es insgesamt? Kennzahlen, Kontostand und P&L-Verlauf für das gewählte Environment (Rundgang durch die App, Abschnitt 2).
- 2 **Analyse → Aktivität** — was ist seit dem letzten Mal passiert? Alle Buchungen über alle Symbole, wochenweise, ohne jede Kampagne einzeln öffnen zu müssen.
- 3 **Kampagnen und Spreads** — bei Bedarf einzelne Positionen genauer ansehen: Verlauf, Buchungen, Ergebnis je Spread (Rundgang, Abschnitte 5 und 6).



4 Log — neue Trades dokumentieren. Die Zahl neben *Log* im Menü zeigt, wie viele Kampagnen der letzten 30 Tage noch keinen Eintrag haben; **Dokumentieren** öffnet den Kommentar zu einer Eröffnung: Plan, Begründung und Ausstiegsszenario, solange sie frisch sind. Die Zahlen kommen aus den Buchungen, und weitere Eröffnungen derselben Kampagne kommentierst du auf deren Seite — so stehen Plan und Ausgang nebeneinander (Rundgang, Abschnitt 7).

5 Psychologie — wie kam der Trade zustande? Stimmung, Motivation, Einhaltung der eigenen Regeln, Versuchungen und die Lektion. Die Regelwerke dafür legst du einmal an; die Auswertung zeigt mit der Zeit, welche Muster Geld kosten (Rundgang, Abschnitt 8).

Log und Psychologie ergänzen sich: Das **Log** hält Plan und Ergebnis **je Trade** fest, die **Psychologie** misst über die Zeit, wie diszipliniert du deinen Regeln folgst.

Tipp

Für den schnellen Eintrag zwischendurch gibt es oben in jeder Seite den gelben Knopf **Neuer Eintrag** (Rundgang durch die App, Abschnitt 1).

3. Kampagnen: offen, wartend, geschlossen

Wie Kampagnen entstehen und enden, beschreibt Überblick — vom Trade bis zur Auswertung, Abschnitt 4.2. Im Alltag heißt das:

- **Aktiv** — es ist noch etwas offen (Aktien oder Optionen).
- **Pending** — alles ist geschlossen, die Kampagne wartet aber noch 10 Tage. Ein neuer Trade im selben Symbol (z. B. der nächste Put einer Wheel-Strategie oder ein Roll) setzt **dieselbe** Kampagne fort.
- **Geschlossen** — 10 Tage ohne neuen Trade. Der nächste Trade in diesem Symbol beginnt eine neue Kampagne.

Eine Kampagne kann also ein paar Tage *Pending* stehen, obwohl die Position längst zu ist — das ist gewollt. Die 10 Tage sind Kalendertage und zählen ab dem Datum des letzten Trades, nicht ab dem Tag, an dem der Import lief (siehe Abschnitt 1, „Muss das täglich laufen?“).



Tipp

Die Frist lässt sich ändern — in der `.env` von `TradingJournal-Weaver`, z. B. `MAX_GAP_DAYS=14` (sinnvoll sind etwa 7 bis 14 Tage). Sie gilt dann für alle künftig verarbeiteten Trades. Bereits geschlossene Kampagnen ändern sich dadurch nicht; wer die neue Frist auch für die Vergangenheit möchte, baut das Environment neu auf (Abschnitt 4).

4. Wenn etwas nicht stimmt

Alle Befehle laufen im PowerShell-Fenster der Desktop-Verknüpfung (Ordner `TradingJournal-Weaver`, Arbeitsumgebung aktiv). Viele Fehlerbilder samt Lösung sammelt außerdem Fehlerbehebung und häufige Fragen.

Eine XML-Datei gefahrlos ausprobieren

Für eine fragliche oder fremde Flex-Query-Datei gibt es das Environment **TEST** — es berührt die echten Daten nicht.

1. **[Windows-Explorer]** Die Datei so benennen, dass sie mit `flex.` beginnt und auf `.xml` endet (z. B. `flex.probe.xml`), und in den Ordner `C:\Trade_XML\test\<benutzer>\incoming\` legen — `<name>` wie im `[[test]]`-Block der `pipeline.toml`.
2. **[PowerShell]** Ganz normal laufen lassen:

```
python run_pipeline.py
```

3. In der App oben auf **TEST** umschalten und das Ergebnis ansehen.

Die Datei wird unabhängig von ihrer Kontonummer immer in TEST eingelesen.

Eine Position bleibt offen, obwohl sie geschlossen ist

Kommt eine Eröffnung an, die passende Schließung aber nie (z. B. wegen einer Datenlücke), bleibt die Position offen stehen. **[PowerShell]** Erst ansehen, was entfernt würde — ohne `--execute` wird nichts gelöscht (Environment, eigenes Präfix und Symbol einsetzen):

```
python -m tools.maintenance.cleanup_stale_positions money --prefix anna_ --symbols XYZ
```

Stimmt die Anzeige, denselben Befehl mit `--execute` am Ende wiederholen. Statt `--symbols` geht auch `--older-than 2026-01-01` (alles vor diesem Datum).



Datenbank sichern und wiederherstellen

Das Skript `db-backup.ps1` sichert **alle** Datenbanken auf einmal: die fünf Environments (mit Trades, Kampagnen, Kontoständen, Trade Log und Psychologie) und die Datenbank der Weboberfläche (Benutzerkonten). Es liest die Zugangsdaten aus der `.env` von TradingJournal-Weaver; MySQL muss laufen.

Sichern — [PowerShell] im Ordner `TradingJournal-Weaver` :

```
.\db-backup.ps1
```

Die Sicherungen landen in `TradingJournal-Weaver\database\db\`, je Datenbank eine Datei `<datenbank>_backup_JJJJMMTT_HHMM.sql`. Mit `-UsbLabel "NAME"` wird zusätzlich auf das USB-Laufwerk mit diesem Namen kopiert (der Name, der im Explorer beim Laufwerk steht).

Wiederherstellen — [PowerShell] im selben Ordner, zuerst als Trockenlauf:

```
.\db-backup.ps1 -Restore -DryRun  
.\db-backup.ps1 -Restore
```

Der Trockenlauf zeigt nur, welche Sicherung je Datenbank eingespielt würde — immer die neueste. Ohne `-DryRun` fragt das Skript für jede Datenbank einzeln nach. Bevor es eine Datenbank überschreibt, sichert es deren aktuellen Stand zusätzlich als `<datenbank>_prerestore_JJJJMMTT_HHMMSS.sql` — ein versehentliches Wiederherstellen lässt sich so rückgängig machen.

Vorsicht

Die Sicherungen enthalten alle Handelsdaten. Nicht weitergeben und nicht in Foren oder auf GitHub hochladen.

Alles neu aufbauen

Wenn die Daten eines Environments durcheinander sind — etwa weil Trades in falscher Reihenfolge ankamen (Abschnitt 1, „Spielt die Reihenfolge der Dateien eine Rolle?“) —, lässt es sich aus der Sammeldatei komplett neu aufbauen.

Achtung

Der Neuaufbau legt die Tabellen des Environments neu an. Dabei gehen auch die Einträge im **Log** und die **Kontostände** verloren. Psychologie-Einträge bleiben erhalten. Vorher die Datenbank sichern (siehe „Datenbank sichern und wiederherstellen“ oben).

[PowerShell] Beispiel für Annas Echtgeld-Bereich:



```
python -m tools.maintenance.reset_IBKR_DB money anna_  
python run_pipeline.py --steps import --from-summary --users anna --envs money  
python run_pipeline.py --steps process --users anna --envs money
```

[Windows-Explorer] Die Kontostände kommen zurück, indem die verarbeiteten CSV-Dateien aus `C:\Trade_CSV\incoming\<benutzer>\processed` wieder nach `C:\Trade_CSV\incoming` kopiert werden; der nächste normale Lauf liest sie ein.

5. Mehrere Benutzer

Jede Person hat eigene Daten (eigenes Präfix, eigene IBKR-Konten in `accounts.toml`). Wie man auf dem eigenen PC zwischen den Benutzern wechselt und wie es auf einem Server aussieht, beschreibt Erster Start — UI-Erstkonfiguration, Abschnitt 3.



Eigener Server

Statt auf dem eigenen PC können TradingJournal-Weaver und TradingJournal auch auf einem eigenen Server laufen — einem gemieteten Windows-Server oder einem Rechner, der dauerhaft eingeschaltet bleibt. Das lohnt sich, wenn du von mehreren Geräten (PC, Notebook, unterwegs) auf **dasselbe** Journal zugreifen möchtest, oder wenn der eigene PC nicht jeden Tag läuft.

Tipp

TradingJournal braucht wenig Leistung. Auf einem Windows-Server von Strato, wie ihn PJM für den automatisierten Handel vorschlägt, läuft es problemlos neben dem übrigen Betrieb — wer einen solchen Server ohnehin hat, braucht keinen zusätzlichen.

Die ausführliche technische Anleitung steht im technischen Handbuch unter Server-Betrieb — TradingJournal auf einem eigenen Server. Dieses Kapitel erklärt, was dabei gleich bleibt und was anders ist.

1. Was gleich bleibt

Die Installation ist im Grunde dieselbe wie auf dem eigenen PC — nur dass du sie per Fernzugriff (Remote Desktop/RDP oder ein ähnliches Programm) direkt auf dem Server ausführst: dieselben Programme (Windows vorbereiten), dieselbe Einrichtung (TradingJournal-Weaver installieren, TradingJournal installieren), dieselben Einstellungsdateien. Auch der Alltag ist derselbe: ein Lauf von `run_pipeline.py` holt und verarbeitet die Daten (Alltag mit dem TradingJournal).



2. Was anders ist

	Eigener PC	Server
Erreichbar	nur auf diesem PC	von allen eigenen Geräten — aber nur über ein VPN (z. B. Tailscale), nie offen im Internet
Webserver	<code>php artisan serve</code> im offenen Fenster, oder Apache	Apache, startet mit Windows und läuft dauerhaft
Anmelden	kein Login, der Admin ist automatisch angemeldet	Login-Seite, jede Person mit eigenem Zugang (<code>APP_ENV=production</code> in der <code>.env</code> von TradingJournal)
Andere Benutzer ansehen	über den Benutzer-Dropdown	nicht möglich, jede Person sieht nur ihre Daten
Start-Environment	TEST	am besten <code>TRADING_DEFAULT_ENV=MONEY</code> in der <code>.env</code> von TradingJournal
Datenlauf	von Hand oder per Aufgabenplanung	per Aufgabenplanung (Abschnitt 4) oder angestoßen von einem anderen Rechner (Abschnitt 3)

Achtung

TradingJournal hat keinen eigenen Schutz gegen das massenhafte Durchprobieren von Passwörtern und ist nicht dafür gebaut, offen im Internet zu stehen. Der Server muss deshalb **nur über ein VPN** erreichbar sein — wie das geht, zeigt die technische Anleitung.

Wie Anmelden und Benutzer auf PC und Server im Detail funktionieren, beschreibt Erster Start — UI-Erstkonfiguration.

3. Wie die Daten auf den Server kommen

Es gibt zwei Wege. Wichtig ist, dass für einen Tag immer **nur ein** Rechner die Daten abholt: Der Mail-Abruf löscht die abgeholten Mails, und die Web-API soll ohnehin höchstens einmal am Tag gefragt werden.

Weg A — der Server holt selbst. Mail-Abruf und Web-API laufen direkt auf dem Server, als Teil des normalen Laufs. Das ist der einfachste Weg; der Lauf startet automatisch per Aufgabenplanung (Abschnitt 4).



Vorsicht

Bei Weg A liegen die **Zugangsdaten** auf dem Server: das Passwort des Postfachs und die IBKR-Prüfcodes für die Web-API stehen dort in der `.env` von TradingJournal-Weaver. Wer das nicht möchte — etwa weil der Server bei einem Anbieter steht —, nimmt Weg B: Dann bleiben die Zugangsdaten auf dem eigenen Rechner, und auf dem Server liegen nur die abgeholten Berichte. Aus diesem Grund war ursprünglich nur Weg B vorgesehen.

Weg B — ein anderer Rechner holt ab und überträgt. Mail-Abruf und Web-API laufen auf einem anderen Rechner, etwa dem eigenen PC oder einem Reise-Notebook. Ein kleines Skript (`push-to-server.ps1`) überträgt die abgeholten Dateien — Flex Queries (XML) und Kontoauszüge (CSV) — per SSH auf den Server und startet dort den Lauf, der sie verarbeitet. Erfolgreich übertragene Dateien verschiebt es auf dem abholenden Rechner in einen Ablageordner (`C:\Trade_XML\pushed\...` , `C:\Trade_CSV\pushed`), damit sie beim nächsten Mal nicht erneut übertragen werden. Auf diesem Rechner braucht es nur eine Minimal-Installation — ohne Datenbank und ohne Weboberfläche. Die Anleitung steht im technischen Handbuch unter Verteilter Abruf — minimaler Client für den Datenabruf.

4. Automatisch laufen lassen (Windows-Aufgabenplanung)

Damit der Lauf jeden Tag von selbst startet — auf dem Server, aber genauso auf dem eigenen PC —, legst du einmal eine Aufgabe in der Windows-Aufgabenplanung an:

1. **[Windows]** Windows-Taste drücken, `Aufgabenplanung` eintippen, öffnen.
2. Rechts **Einfache Aufgabe erstellen...** — Name z. B. `TradingJournal Pipeline` .
3. Trigger **Täglich**, Uhrzeit z. B. **18:00**. Die Mails zu einem Handelstag kommen am folgenden Nachmittag (IBKR einrichten — Flex Queries und Kontoauszug, Abschnitt 4); ein Lauf am frühen Abend erwischt sie zuverlässig.
4. Aktion **Programm starten**, dann eintragen (Pfade an die eigene Installation anpassen):

Feld	Inhalt
Programm/Skript	<code>C:\Users\<dein Anmeldenamen>\trading\TradingJournal-Weaver\.env\Scripts\python.exe</code>
Argumente hinzufügen	<code>run_pipeline.py</code>
Starten in	<code>C:\Users\<dein Anmeldenamen>\trading\TradingJournal-Weaver</code>

5. Vor **Fertig stellen** den Haken bei **Beim Klicken auf „Fertig stellen“ die Eigenschaften für diese Aufgabe öffnen** setzen. In den Eigenschaften auf der Registerkarte **Allgemein Unabhängig von der Benutzeranmeldung ausführen** wählen und mit **OK** bestätigen (fragt nach dem Windows-



Passwort).

Ob es geklappt hat: In der Aufgabenplanung zeigt die Spalte *Letztes Ausführungsergebnis* nach dem ersten Lauf `(0x0)`; in der App sind die neuen Daten da. Zum Ausprobieren lässt sich die Aufgabe per Rechtsklick → **Ausführen** sofort starten.

5. Updates

Ein Update läuft auf dem Server genauso wie auf dem PC: per Fernzugriff verbinden und im Ordner `TradingJournal` `.\update.ps1` starten (Updates einspielen). Holt bei Weg B ein anderer Rechner die Daten ab, dort im Ordner `TradingJournal-Weaver` zusätzlich `git pull` ausführen, damit auch der Abruf auf dem neuesten Stand ist.

Updates einspielen

Wie eine bestehende Installation — auf dem eigenen PC oder auf einem Server — auf den neuesten Stand von TradingJournal-Weaver und TradingJournal kommt. Für die Erstinstallation siehe TradingJournal-Weaver installieren und TradingJournal installieren.

Was ein Update macht — und was nicht

Ein Update holt den neuen Code von GitHub und erledigt alles, was danach nötig ist, damit er läuft: neue Programmbausteine (Python, PHP, JavaScript) installieren, kleine Anpassungen an den Datenbank-Tabellen vornehmen, die Weboberfläche neu bauen.

Deine Daten bleiben unangetastet: Trades, Kampagnen, Log, Psychologie und Kontostände werden nicht neu aufgebaut, `reset_IBKR_DB.py` ist für ein Update nie nötig. Ebenso bleiben deine eigenen Einstellungsdateien unverändert — die beiden `.env`-Dateien, `users.toml`, `accounts.toml` und `pipeline.toml`.

Update starten

Das Skript `update.ps1` im Ordner `TradingJournal` erledigt alle Schritte in der richtigen Reihenfolge — für **beide** Programme.

1. Vorher sicherstellen, dass gerade **kein Lauf** von `run_pipeline.py` aktiv ist.
2. **[PowerShell]** in den Ordner `TradingJournal` wechseln (Pfad an die eigene Installation anpassen) und das Skript starten:

```
cd C:\Users\<dein AnmeldeName>\trading\TradingJournal
.\update.ps1
```

3. Am Ende steht in grün `Update abgeschlossen: ...`. Danach im Browser die Seite neu laden.

Voraussetzung: `TradingJournal-Weaver` und `TradingJournal` liegen nebeneinander im selben Ordner — so wie bei der Installation. Meldet PowerShell, dass das Ausführen von Skripten deaktiviert ist, hilft derselbe einmalige Schritt wie beim Einrichten der Desktop-Verknüpfung (TradingJournal-Weaver installieren, Schritt 7).

Neu starten muss man danach nichts — weder `php artisan serve` noch Apache. Die Weboberfläche liest den neuen Stand beim nächsten Seitenaufruf.



Auf einem Server genauso, nur vorher per Fernzugriff (RDP o. ä., siehe Technisches Handbuch Server-Betrieb — TradingJournal auf einem eigenen Server) auf den Server verbinden und dort im Ordner `TradingJournal` `.\update.ps1` starten.

Was das Skript im Einzelnen tut

- 1 **Code holen** — `git pull` in `TradingJournal-Weaver` und in `TradingJournal`
- 2 **Python-Pakete** für TradingJournal-Weaver aktualisieren (`pip install -r requirements.txt` in dessen Arbeitsumgebung)
- 3 **Datenbank-Anpassungen** — die Skripte `TradingJournal-Weaver/dev/migrate_*.py` ausführen; sie ergänzen fehlende Spalten oder Tabellen und überspringen, was schon da ist
- 4 **PHP- und JavaScript-Pakete** installieren und die Weboberfläche neu bauen (`composer install`, `npm install`, `npm run build`)
- 5 **Laravel-Tabellen** aktualisieren (`php artisan migrate --force`, z. B. die Benutzer)
- 6 **Zwischenspeicher** von Laravel leeren und neu aufbauen
- 7 **Versionsstand** notieren, damit die App ihn anzeigen kann (siehe unten)

Geht ein Schritt schief, **bricht das Skript sofort ab** und sagt, wo — statt mit einem halb aktualisierten Stand weiterzumachen. Einzige Ausnahme sind die Datenbank-Anpassungen aus Schritt 3: Sie melden ein Problem als gelbe Warnung, und das Skript läuft weiter — die Warnung dann trotzdem ernst nehmen. Nach dem Beheben einfach `.\update.ps1` noch einmal starten; alle Schritte lassen sich gefahrlos wiederholen.



Nach dem Update: neue Einstellungen?

Neue Versionen bringen manchmal zusätzliche, **optionale** Einstellungen mit (z. B. `MAX_GAP_DAYS` für die Kampagnen-Frist). Sie stehen in den Vorlagen `.env.example` der beiden Programme, landen aber nicht automatisch in der eigenen `.env`. Wer eine neue Einstellung nutzen möchte, überträgt die Zeile von Hand. Ohne Eintrag gilt jeweils der bisherige Standard.

Sicherheitsupdates für PHP

`update.ps1` aktualisiert nur TradingJournal und TradingJournal-Weaver, nicht die Programme darunter. PHP kommt bei einer XAMPP-Installation mit XAMPP mit — und XAMPP liefert neue PHP-Versionen oft erst Wochen nach einem Sicherheitsupdate aus. Wird eine kritische PHP-Lücke bekannt (Ankündigungen unter [php.net](https://www.php.net/news) → News), lässt sich PHP auch selbst austauschen, ohne auf ein neues XAMPP zu warten. Die Schritt-für-Schritt-Anleitung steht im technischen Handbuch (Installation: Windows-Grundausstattung, Abschnitt „PHP aktualisieren, ohne auf ein neues XAMPP zu warten“).

Welche PHP-Version gerade läuft — **[PowerShell]**:

```
php -v
```

Welche Version läuft gerade?

Unten in der Seitenleiste der App steht **Build**: mit einer kurzen Kennung (z. B. `Build: e1db712`); fährt man mit der Maus darüber, erscheinen Versionsbezeichnung und Zeitpunkt des Updates. Die Anzeige erscheint erst nach dem **ersten** Lauf von `update.ps1` — direkt nach der Installation ist dort noch nichts zu sehen.

Ohne App nachsehen — **[PowerShell]** im Ordner `TradingJournal`:

```
git describe --tags --always
```



Wenn etwas schiefgeht

Problem	Ursache	Lösung
<code>git pull</code> schlägt fehl	Im Ordner wurden Dateien geändert, die auch das Update ändert, oder es liegt eine gleichnamige neue Datei im Weg	[PowerShell] im genannten Ordner <code>git status</code> – eigene Änderungen sichern (<code>git stash</code>) oder die störende Datei umbenennen, dann <code>.\update.ps1</code> erneut starten
<code>pip install</code> , <code>composer install</code> oder <code>npm install</code> schlägt fehl	Fehlendes Programm bzw. fehlende PHP-Erweiterung, oder keine Internetverbindung	Die Fehlermeldung nennt meist die Ursache; Grundausrüstung prüfen (Windows vorbereiten)
<code>php artisan migrate --force</code> meldet einen Fehler	Die Datenbank wurde von Hand verändert und passt nicht zur erwarteten Struktur	Fehlermeldung lesen; vor einem weiteren Versuch die Datenbank sichern (Alltag mit dem TradingJournal, „Datenbank sichern und wiederherstellen“)
Die Seite zeigt nach dem Update noch den alten Stand	Der Browser zeigt eine zwischengespeicherte Fassung	Seite mit Strg + Shift + R neu laden; prüfen, ob <code>npm run build</code> ohne Fehler durchlief
Ein Update verlangt eine neue Python- oder PHP-Version bzw. -Erweiterung	Das kann das Skript nicht selbst installieren	Wie bei der Ersteinrichtung nachinstallieren (Windows vorbereiten), dann <code>.\update.ps1</code> erneut starten

Fehlerbehebung und häufige Fragen

Dieses Kapitel sammelt Probleme, die bei Einrichtung und Betrieb auftreten können, mit Ursache und Lösung. Es wächst mit den Fragen aus der Praxis — wer auf ein Problem stößt, das hier fehlt, meldet es gerne: am besten **im Forum**, in dem das TradingJournal vorgestellt wurde. Fragen, die öfter kommen, werden dort gesammelt und zu einer FAQ zusammengefasst. Wer ein GitHub-Konto hat, kann stattdessen ein Issue anlegen — mit den Vorlagen „Fehler melden“ bzw. „Idee / Wunsch“ im Repository [TradingJournal](#) oder [TradingJournal-Weaver](#).

Achtung

Bei Fragen in Foren oder auf GitHub **niemals** echte Kontonummern, Flex-Query-Dateien, Prüfcodes (Token) oder unmaskierte Screenshots weitergeben. Für die Fehlersuche reicht fast immer die Fehlermeldung aus dem Protokoll.

1. Zuerst nachsehen: die Protokolle

Die meisten Probleme lassen sich an einer Meldung in den Protokolldateien erkennen:

Programm	Protokolldatei
TradingJournal-Weaver	<code>TradingJournal-Weaver\log\app.log</code>
TradingJournal (Weboberfläche)	<code>TradingJournal\storage\logs\laravel.log</code>

Dazu kommt im Journal selbst die Seite **Raw Data** (unter *Analyse*): Steht ein Trade dort, ist er angekommen und das Problem liegt in Verarbeitung oder Anzeige. Fehlt er, liegt es vor dem Import — beim Bericht, bei der Mail oder bei der Kontozuordnung.

2. Installation

„Fatal error in launcher: Unable to create process using ...“

Tritt auf, nachdem der Projektordner verschoben oder umbenannt wurde. Die virtuelle Python-Umgebung (`.venv`) enthält feste Pfade und funktioniert am neuen Ort nicht mehr. Lösung: die Umgebung neu anlegen.



PowerShell, im Ordner `TradingJournal-Weaver` :

```
Remove-Item -Recurse -Force .venv
python -m venv .venv
.venv\Scripts\activate
pip install -r requirements.txt
```

„Vite manifest not found at: ... \public\build\manifest.json“

Die Weboberfläche zeigt diese Fehlerseite, wenn Farben, Icons und Layout noch nicht gebaut wurden. Lösung: im Ordner `TradingJournal` einmal `npm run build` ausführen (Kapitel TradingJournal installieren, Schritt „Optik der Seite bauen“). Beim Einspielen von Updates mit `update.ps1` geschieht das automatisch.

„UnicodeDecodeError: 'utf-8' codec can't decode byte 0xef ...“ (oder 0xff) beim Start

Eine Konfigurationsdatei wie `pipeline.toml` oder `accounts.toml` wurde mit einer unpassenden Kodierung gespeichert — typischerweise, wenn sie per PowerShell (`Set-Content` , `>`) erzeugt wurde. Lösung: die Datei im Text-Editor öffnen und mit der Kodierung **UTF-8** (nicht „UTF-8 mit BOM“) neu speichern.

Nach dem Login: „Diese Seite funktioniert nicht — zu oft weitergeleitet“

Meist kann die Weboberfläche die Trading-Datenbank nicht erreichen. In `storage\logs\laravel.log` nach `ERROR` suchen. Häufige Auslöser: `TRADING_DB_USERNAME` oder `TRADING_DB_PASSWORD` in der `.env` falsch, oder für den Benutzer gibt es in keinem Environment Tabellen — dann `reset_IBKR_DB.py` mit genau dem Präfix ausführen, der beim Anlegen des Benutzers angegeben wurde (TradingJournal-Weaver installieren, Schritt 4).

3. Daten abholen (Mail und Web-API)

Es werden keine Berichte aus dem Postfach geholt

- IBKR schickt die Berichte **nur an die E-Mail-Adresse, die beim Anlegen des IBKR-Kontos hinterlegt wurde**. `IMAP_USERNAME` in der `.env` muss genau dieses Postfach sein (oder eines, in das weitergeleitet wird).
- Gelesen wird nur der **Posteingang**. Verschiebt eine Filterregel die IBKR-Mails in einen anderen Ordner, findet der Abruf sie nicht.
- Im Protokoll steht „IMAP_USERNAME/IMAP_PASSWORD nicht konfiguriert oder noch auf Platzhalterwert“: Die Zugangsdaten in der `.env` sind leer oder noch die Vorlage.

Web-API: „NameResolutionError ... Failed to resolve ,gdcdyn.interactivebrokers.com““

Kein IBKR-Problem, sondern ein Namensauflösungs-Fehler (DNS) im eigenen Netz — häufig an Handy-Hotspots oder kleinen Routern. Der Mail-Abruf ist davon nicht betroffen. Lösung: feste DNS-Server eintragen.



PowerShell **als Administrator** (`WLAN` durch den Namen der eigenen Netzwerkverbindung ersetzen):

```
Set-DnsClientServerAddress -InterfaceAlias "WLAN" -ServerAddresses 1.1.1.1, 8.8.8.8  
ipconfig /flushdns
```

Rückgängig machen mit `Set-DnsClientServerAddress -InterfaceAlias "WLAN" -ResetServerAddresses`.

Die Einstellung bleibt bei Netzwechsel und Neustart erhalten, gilt aber nur für diese eine Netzwerkverbindung (WLAN und Netzkabel getrennt). Kommt der Fehler wieder, zuerst prüfen, ob sie noch gesetzt ist — PowerShell (ohne Administrator):

```
Get-DnsClientServerAddress -InterfaceAlias "WLAN" -AddressFamily IPv4
```

Steht dort nicht `1.1.1.1, 8.8.8.8`, die Einstellung erneut setzen. Erscheint in einem offenen WLAN (Hotel, Bahn) die Anmeldeseite nicht, die Einstellung kurz zurücksetzen, anmelden und danach wieder setzen.

Web-API: Fehler 1025 („Too many failed attempts“)

IBKR sperrt den Abruf vorübergehend, wenn kurz hintereinander zu viele fehlgeschlagene Anfragen kommen. Den Abruf dann **nicht** wiederholt neu starten — das verlängert die Sperre. Einige Zeit warten und vorher prüfen, ob Prüfcode (Token) und Query-ID in `IBKR_FLEX_QUERIES` stimmen. Die Daten kommen in der Zwischenzeit weiter per Mail.

Web-API: „Daten noch nicht verfügbar nach 5 Versuchen“

IBKR erstellt den Bericht noch. Das kommt kurz nach Handelsschluss vor; einfach später erneut abrufen.

4. Import

Trades fehlen — im Protokoll: „Unbekannte accountId ,...‘ in ... — übersprungen“

Das Konto ist in `accounts.toml` nicht eingetragen. Die Zuordnung läuft über die **letzten vier Ziffern** der IBKR-Kontonummer. Nach dem Eintragen den Import erneut laufen lassen; die übersprungenen Dateien liegen noch im Eingangsordner.

Im Protokoll: „Konnte Datum nicht parsen: ...“

Die Flex Query verwendet ein anderes Datumsformat. Pflicht ist `dd/MM/yyyy`, Zeit `HHmmss`, Trennzeichen `;` (Kapitel IBKR einrichten, allgemeine Konfiguration).



Kontostand, Rendite und Max Drawdown bleiben leer („Keine NAV-Daten vorhanden“)

Diese Werte kommen ausschließlich aus dem täglichen **Kontoauszug (CSV)**, nicht aus der Flex Query. Prüfen:

- Ist der Kontoauszug bei IBKR eingerichtet und wird er per Mail zugestellt?
- Ist als **Sprache English** eingestellt? Der Import erkennt die Abschnitte an ihren englischen Bezeichnungen.
- Steht im Protokoll „accountId ‚...‘ nicht in accounts.toml — übersprungen“? Dann fehlt das Konto in `accounts.toml` (wie oben).

5. Anzeige im Journal

Neue Trades sind nicht zu sehen

Der Reihe nach prüfen:

1. Ist die Pipeline seit dem Trade gelaufen? IBKR erstellt die Berichte erst nach Handelsschluss — ein bis zwei Tage Nachlauf sind normal.
2. Ist oben das richtige **Environment** gewählt?
3. Schließt der **Zeitraum**-Filter den Trade aus? Testweise auf „Alle“ stellen.
4. Steht der Trade unter **Raw Data**? Wenn nicht, liegt das Problem beim Abholen oder Import (Abschnitte 3 und 4).

Eine Position bleibt offen, obwohl sie bei IBKR geschlossen ist

Der schließende Trade ist nie angekommen, etwa weil an diesem Tag weder Mail noch Web-API geliefert haben. Zuerst unter **Raw Data** nachsehen. Fehlt der Trade dort, den Bericht für den betreffenden Tag erneut abrufen. Für endgültig liegengebliebene Positionen gibt es ein Wartungsskript (Alltag mit dem Trading Journal, Abschnitt 4 „Eine Position bleibt offen, obwohl sie geschlossen ist“).

Warum weichen die Gewinne vom IBKR-Kontoauszug ab?

Das Journal zeigt P&L überall **brutto** (ohne Kommissionen) und den **Netto**-Wert darunter. Für den Vergleich mit IBKR den Netto-Wert verwenden.