

# Projekt: Aktienbewertungen-App (lokal, ohne Login)

---

## Ziel der App

Du willst eine private App, mit der du:

- automatisch Kursdaten und Kennzahlen zu Aktien abrufst
  - auf Basis deiner Formel selbst den **fairen Wert** berechnest
  - erkennst, ob eine Aktie über- oder unterbewertet ist
  - **alles dauerhaft speicherst**, ohne jedes Mal neu eintragen zu müssen
- 

## Was die App konkret macht

### Automatisch Aktieninfos holen

Du gibst z. B. „Apple“ oder das Kürzel **AAPL** ein. Die App holt sich selbstständig:

- aktuellen Aktienkurs
- Gewinn pro Aktie (EPS)
- KGV (Kurs-Gewinn-Verhältnis)
- Gewinnwachstum usw.

### Selbst rechnen

Die App rechnet automatisch:

- wie viel die Aktie wert sein sollte (deine Formel)
- wie viel Potenzial noch im Kurs steckt
- ob die Aktie **unterbewertet (grün)** oder **überbewertet (rot)** ist

### Alles übersichtlich anzeigen

- In einer Tabelle wie auf deinem Screenshot

- Mit farblicher Bewertung: grün = unterbewertet / rot = überbewertet

### ✓ Neue Aktien eintragen

- Du kannst jederzeit neue Aktien hinzufügen
- Die App holt sich automatisch die nötigen Infos

### ✓ Daten bleiben bei dir

- Alles läuft **nur auf deinem Rechner**
- Keine Daten werden online gespeichert



## Was die App kann (Funktionen)

#	Funktion	Beschreibung
1	Aktie hinzufügen	Du gibst z. B. das Symbol „AAPL“ ein
2	Kursdaten laden	App holt Kurs, EPS, KGV, Wachstum von einer API (z. B. Finnhub)
3	Berechnung	App berechnet fairen Wert + Potenzial nach deiner Formel
4	Tabelle anzeigen	Übersicht mit Farben (grün = unterbewertet, rot = überbewertet)
5	Daten speichern	Alles wird lokal in einer SQLite-Datenbank gespeichert
6	Bearbeiten / Löschen	Du kannst Aktiendaten nachträglich ändern oder entfernen



## Wie die App deine Daten speichert

### Was wird gespeichert?

Für jede Aktie:

- Name (z. B. Apple, Tesla)
- Symbol (z. B. AAPL, TSLA)

- Aktueller Kurs
- Gewinn pro Aktie
- KGV
- Gewinnwachstum (5 Jahre)
- Fairer Wert (ausgerechnet)
- Potenzial in %
- Kommentar (optional)
- Letztes Update

### Welche Datenbank wird verwendet?

- **SQLite**
- Lokale Datei (z. B. `stocks.db`)
- Kein Server, keine Installation nötig
- Wird direkt von der App erstellt und genutzt

### Was bringt das?

- Du kannst deine Liste jederzeit erweitern
- Alles bleibt gespeichert – auch nach einem Neustart
- Später kannst du Suchfunktionen, Sortierung, Filter einbauen
- Oder Daten als Excel/CSV exportieren

---

### Was gespeichert wird (Datenbankfelder)

- `id`
- `name`
- `symbol`
- `current_price`

- `eps`
- `pe_ratio`
- `growth_5y_percent`
- `eps_in_5y`
- `target_pe_ratio`
- `fair_value`
- `price_diff`
- `potential_percent`
- `last_updated`

## Technologie-Stack (einfach gehalten)

Bereich	Tool	Erklärung
Backend	FastAPI	Verarbeitet Anfragen, rechnet & ruft Daten ab
Frontend	HTML + JS	Zeigt Tabelle & Eingabeformular
Datenbank	SQLite	Lokale Datei, kein Setup nötig
API	Finnhub.io	Holt Kurs, EPS, KGV usw. automatisch

---












## Aufwandsschätzung

Aufgabe	Dauer
Projektgrundgerüst + Setup	1 Stunde
Datenbankmodell + Verbindung	1 Stunde
API-Anbindung (z. B. Finnhub)	2 Stunden
Formelberechnung & Logik	2 Stunden
Web-Oberfläche (HTML + JS)	3 Stunden
Farbliche Bewertung & Sortierung	1 Stunde
Testen & Feinschliff	1–2 Stunden
<b>Gesamt</b>	<b>10–12 h</b>

---



## Mögliche Erweiterungen für später

Bereich	Idee / Feature
 Login	Benutzerverwaltung mit Login (z. B. für mehrere Nutzer oder Cloud-Version)
 Hosting	App im Internet verfügbar machen (z. B. über Fly.io, Render oder Railway)
 Mehr Analysen	Weitere Finanzkennzahlen integrieren (z. B. Umsatz, Verschuldung, Dividende)
 Export	Daten als Excel oder CSV exportieren
 Import	Daten aus bestehenden Tabellen importieren
 Filter & Sortierung	Aktien nach Bewertung, Potenzial oder Name filtern und sortieren
 Alerts	Warnungen bei starken Kursschwankungen oder Bewertungsänderungen
 KI-Integration	Automatisierte Bewertung auf Basis von Analystenmeinungen & News
 Strategie-Scoring	Bewertung nach Graham, Buffett oder eigenen Regeln
 Diagramme	Erweiterte Charts und Visualisierungen (Kursverlauf, Vergleich etc.)
 Modul-System	Eigene Bewertungsmodelle modular hinzufügen

## Custom instructions

### What would you like ChatGPT to know about you to provide better responses?

I am building a personal stock valuation app using FastAPI, HTML/JS, and SQLite.

I want the app to retrieve stock data (e.g. price, EPS, P/E ratio, growth) via an API like Finnhub, calculate fair value based on my own formula, and show the result in a colored table (e.g. green = undervalued).

I'm storing everything locally in a SQLite database, no user login is required.

I'm comfortable with HTML/CSS and basic Python, but I want clean, well-structured guidance.

---

### How would you like ChatGPT to respond?

Give me full code examples when needed, but keep them clean and modular.

Explain database models, API calls, and logic clearly but without unnecessary theory.

Use only Python (FastAPI), HTML, JavaScript and SQLite – no extra frameworks or libraries unless needed.

Keep everything focused on building the actual app step by step.

When giving multiple options, recommend the one that is simpler or more stable.

# Projekt-Outline: Persönliche Aktienbewertungs-App (lokal)

## Modul 1: Projekt-Setup 🚀📁⚙️






- 📁 Ordnerstruktur erstellen  
(Das bedeutet: Du erstellst einen Projektordner mit Unterordnern, z. B.:
  - `backend/` für FastAPI-Dateien,
  - `frontend/` für HTML und JavaScript,
  - `data/` für deine SQLite-Datenbankdatei und eventuell spätere CSV-Dateien.)
- ⚙️ FastAPI-Projekt initialisieren  
(FastAPI ist dein Mini-Server für:
  - Schnittstelle zwischen deinPython-Code und Website. Berechnung deiner Formel,
  - Speicherung der Daten in SQLite,
  - Übergabe der Daten an die Webseite zur Anzeige.)

Finnhub = Datenlieferant, FastAPI = Steuerzentrale deiner App.
- 🗄️ SQLite-Datei vorbereiten  
(Du legst eine SQLite-Datei wie z. B. "stocks.db" an, die später alle Aktieninformationen speichert. Sie wird lokal auf deinem Rechner erstellt und dauerhaft genutzt, um z. B. Kurs, EPS und Bewertungen zu speichern. FastAPI liest und beschreibt diese Datei.)



## Modul 2: Datenbankmodell 🗄️🔧🧱

- 🗄️ SQLite-Datenbankstruktur definieren  
(Das bedeutet: Du legst fest, welche Felder deine Tabelle haben soll. Also z. B.: Name der Aktie, Symbol (wie AAPL für Apple), Kurs, EPS, KGV, Fairer Wert usw. Diese Struktur nennt man auch „Schema“.)
- 🐸 Python-Modell für Aktien anlegen (models.py)  
(Das heißt: Du erstellst in einer Datei namens "models.py" eine Python-Klasse, die beschreibt, wie eine Aktie in der Datenbank gespeichert wird. Die Klasse enthält Felder wie Name, Symbol, Kurs, EPS usw. Sie ist notwendig, damit FastAPI weiß, wie man Aktien speichert und abruft.)
- 🔌 Datenbankverbindung mit FastAPI herstellen  
(Das bedeutet: Du richtest in FastAPI die Verbindung zur SQLite-Datenbank ein. Das passiert meistens in einer Datei wie `main.py` oder `database.py`. Du verwendest dort `sqlite3` oder eine ORM wie `SQLModel`, um die Datenbank zu öffnen und Anfragen an sie zu senden.)




## Modul 3: API-Anbindung (z. B. Finnhub) 🌐🔗🛠️

-  API-Key vorbereiten  
(Der API-Key ist ein persönlicher Zugangsschlüssel, den du kostenlos auf der Website von Finnhub bekommst. Damit kannst du Daten abrufen. Für Hobby- und Einzelentwickler ist der Basiszugang bei Finnhub in der Regel kostenlos.)
-  Abruf von Kurs, EPS, KGV, Wachstum  
(Das machen wir mit einer Funktion in **Python**. Diese Funktion schickt eine Anfrage an die Finnhub-API. Du gibst z. B. das Symbol 'AAPL' an, und die Funktion holt die nötigen Daten. Die API liefert sie im JSON-Format zurück, z. B. `current_price`, `eps`, `pe_ratio`, `growth`. Diese Daten speichern wir dann in der Datenbank und nutzen sie für die Bewertung. JavaScript brauchen wir dafür nicht – das passiert alles im Backend mit Python.)
-  Hilfsfunktionen schreiben (`fetch_stock_data`)  
(Das bedeutet: Du programmierst in Python eine eigene Funktion – z. B. `fetch_stock_data(symbol)` – die alle nötigen Daten zu einer Aktie von der API holt. Diese Funktion kümmert sich darum, eine Verbindung zur Finnhub-API herzustellen, die Daten als JSON zu empfangen und daraus die Infos wie Kurs, EPS, KGV usw. zu extrahieren. Danach kannst du diese Daten weiterverarbeiten oder direkt in der Datenbank speichern.)  
**Achtung:**
- „ Abruf“ beschreibt *was* wir tun wollen: Kursdaten besorgen.
- „ Hilfsfunktion“ beschreibt *wie* wir das technisch lösen – nämlich mit einer selbstgeschriebenen Python-Funktion.

#### Modul 4: Berechnung der Bewertung

-  Bewertungsformel umsetzen  
(Das bedeutet: Du programmierst in Python eine Funktion, die deine persönliche Formel zur Aktienbewertung umsetzt. Diese Funktion nimmt z. B. Kurs, EPS, Wachstum usw. als Eingabe und berechnet daraus:
  - den fairen Wert,
  - das Kurs-Potenzial,
  - eine Bewertung (z. B. unterbewertet oder überbewertet).)
-  Felder berechnen: fairer Wert, Potenzial, Bewertung





#### Modul 5: Daten speichern & verwalten

-  Neue Aktien speichern  
(Mit einer Python-Funktion speicherst du neue Aktien in der SQLite-Datenbank. Dafür verwendest du SQL-Befehle oder ein ORM wie SQLAlchemy. Diese Funktion wird über einen FastAPI-Endpunkt angesprochen.)
-  Aktualisieren und  Löschen ermöglichen  
(Auch das passiert über Python: Du erstellst zwei weitere Funktionen für "Update" und "Delete". Diese werden über entsprechende FastAPI-Routen aufgerufen. Tools,

die du dafür brauchst: Python, FastAPI, SQLite/SQLModel.)



## Modul 6: Web-Oberfläche (HTML + JS)

(Dieser Teil wird mit **HTML und JavaScript** umgesetzt. JavaScript wird benötigt, um:

- das Eingabeformular mit Leben zu füllen (z. B. wenn du auf "Speichern" klickst),
- die JSON-Daten vom Backend (FastAPI) abzurufen,
- die Tabelle mit Aktien dynamisch darzustellen,
- Farben wie grün/rot für Bewertungen je nach Inhalt anzuzeigen.  
Du brauchst dafür keine speziellen Tools – ein Texteditor und ein Browser genügen. Das Frontend sendet Eingaben als JSON an FastAPI und zeigt Daten aus der Datenbank in einer Tabelle an.)
-  Eingabeformular für neue Aktien
-  Tabelle zur Darstellung
-   Farben für Bewertung (grün/rot)




## Modul 7: Backend-Routen (FastAPI)

(Diese Routen erstellst du in **Python** mit FastAPI. Du definierst URLs wie `/add`, `/list`, `/update`, `/delete`, die vom Frontend aufgerufen werden. Die Kommunikation erfolgt über JSON.)

-  Endpunkte definieren: `/add`, `/list`, `/update`, `/delete`
-  JSON-Antworten für das Frontend bereitstellen

## Modul 8: Tests & Feinschliff

(Alle Tests und Prüfungen führst du mit **Python** durch. Du kannst einfache `assert`-Anweisungen oder Tools wie `pytest` nutzen. Damit testest du z. B. ob die Berechnung richtig ist oder ob Eingaben korrekt behandelt werden.)

-  Datenbanktests
-  Eingabefehler abfangen
-  Berechnungen verifizieren

Bonus (optional)   



(Alle Erweiterungen lassen sich mit den bestehenden Tools umsetzen: Python für Logik, FastAPI für Schnittstellen, HTML/JS für Darstellung. Filter & Sortierung z. B. mit JavaScript im Frontend oder direkt bei der Datenabfrage in Python.)

-  Filter & Sortierung
-  Datenexport als CSV
-  Weitere Kennzahlen