Real Time Stock Tracker

By Arnel Veladzic, Ivan Knöfler, Fabio Gomes Silva

Inhalt

- Idee
- Anforderungen / Technologien
- Live Demo
- Architektur
- Monorepo
- Components
- Design Decisions
- Fazit Modul
- Q&A

Idee

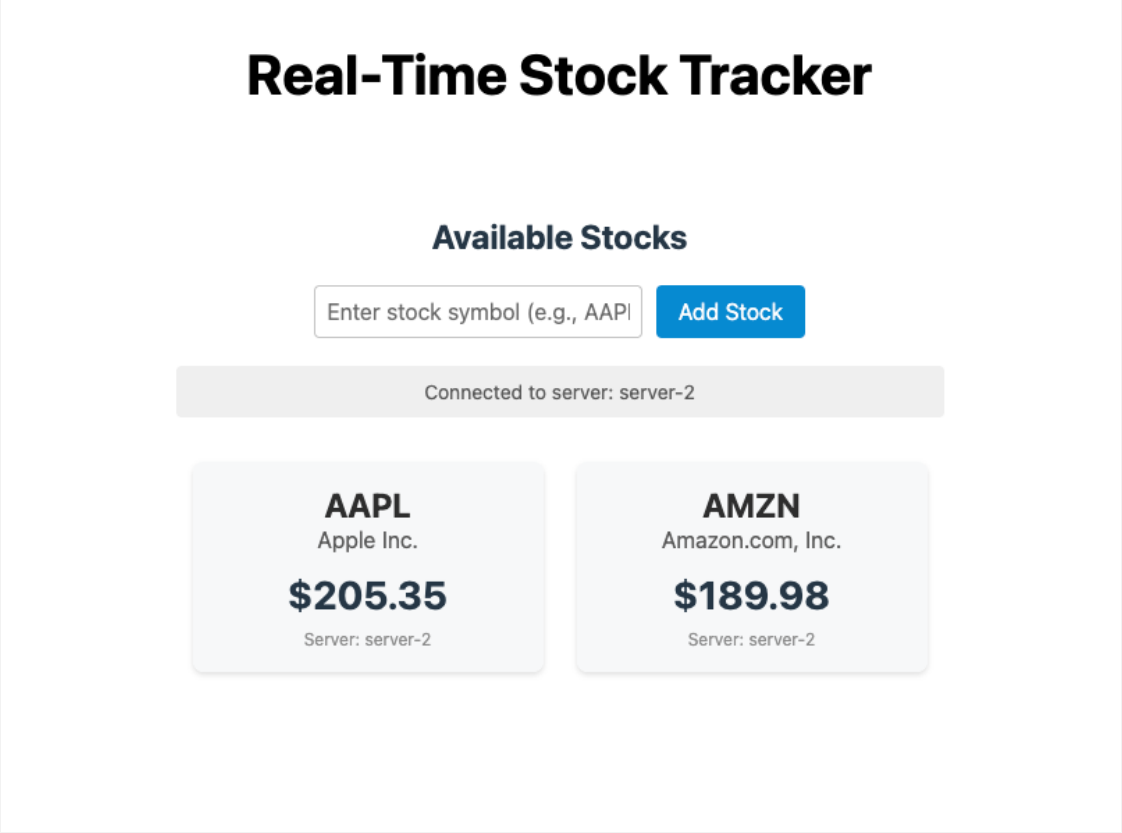
➢ A real-time distributed stock price tracker

1. **Clients subscribe** to stock price updates via SSE.
2. The **backend fetches** stock prices and pushes updates to connected clients.
3. **Traefik load balances** requests between two backend instances.
4. If one backend **fails**, the other instance continues serving requests.
5. **Stock price history** is stored in PostgreSQL for later analysis.

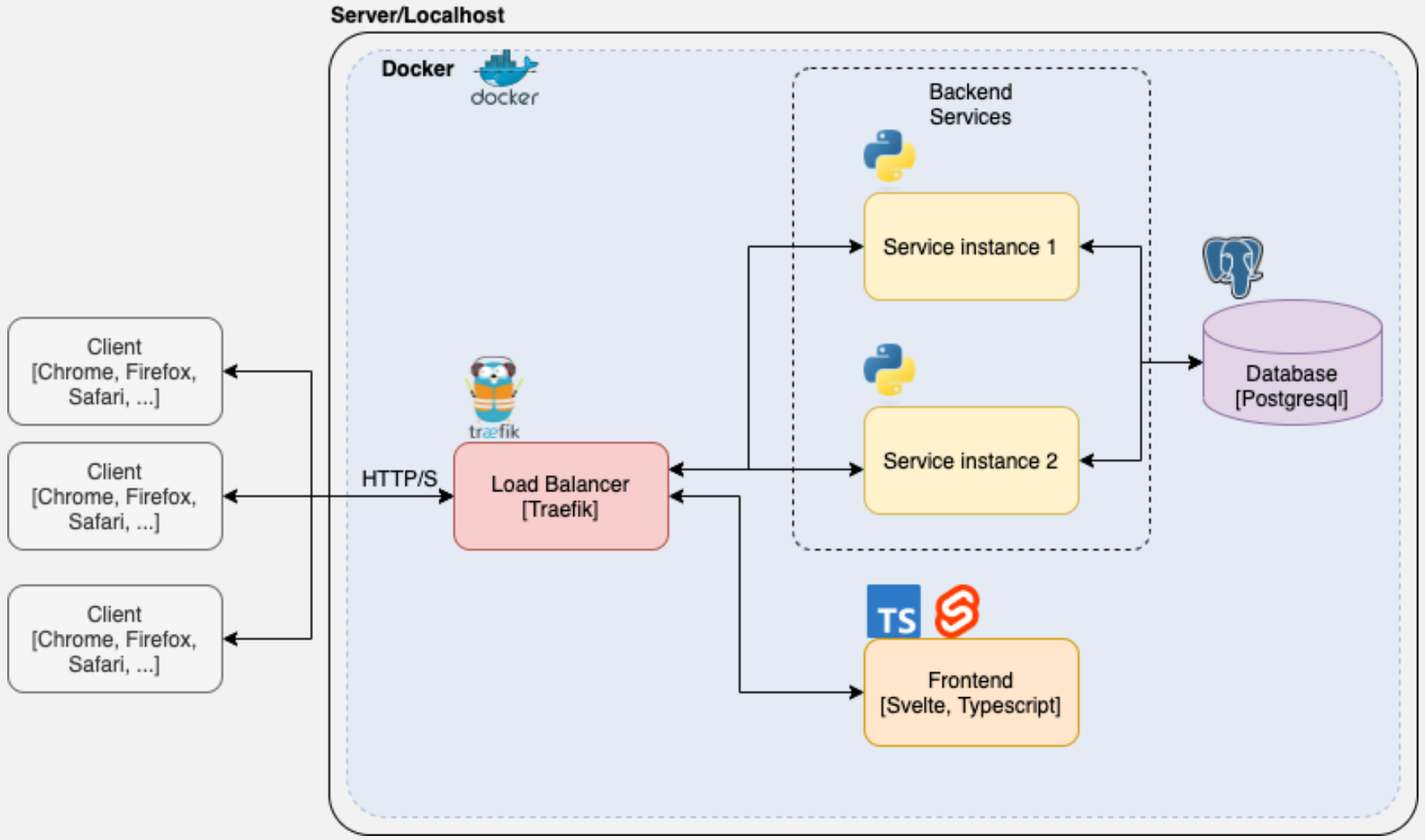
Anforderungen

Simple Frontend	➢ Svelte, Typescript, Vite
Two instances of a service (with shutdown demo)	➢ 2x Python Instanzen als Backend
Loadbalancing with multiple instances	➢ Traefik
Database (persistent storage)	➢ PostgreSQL
Dockerized	➢ Done
SSE sent from server to the client	➢ Done
Use latest stable releases of chosen libraries and frameworks	➢ Done

Demo

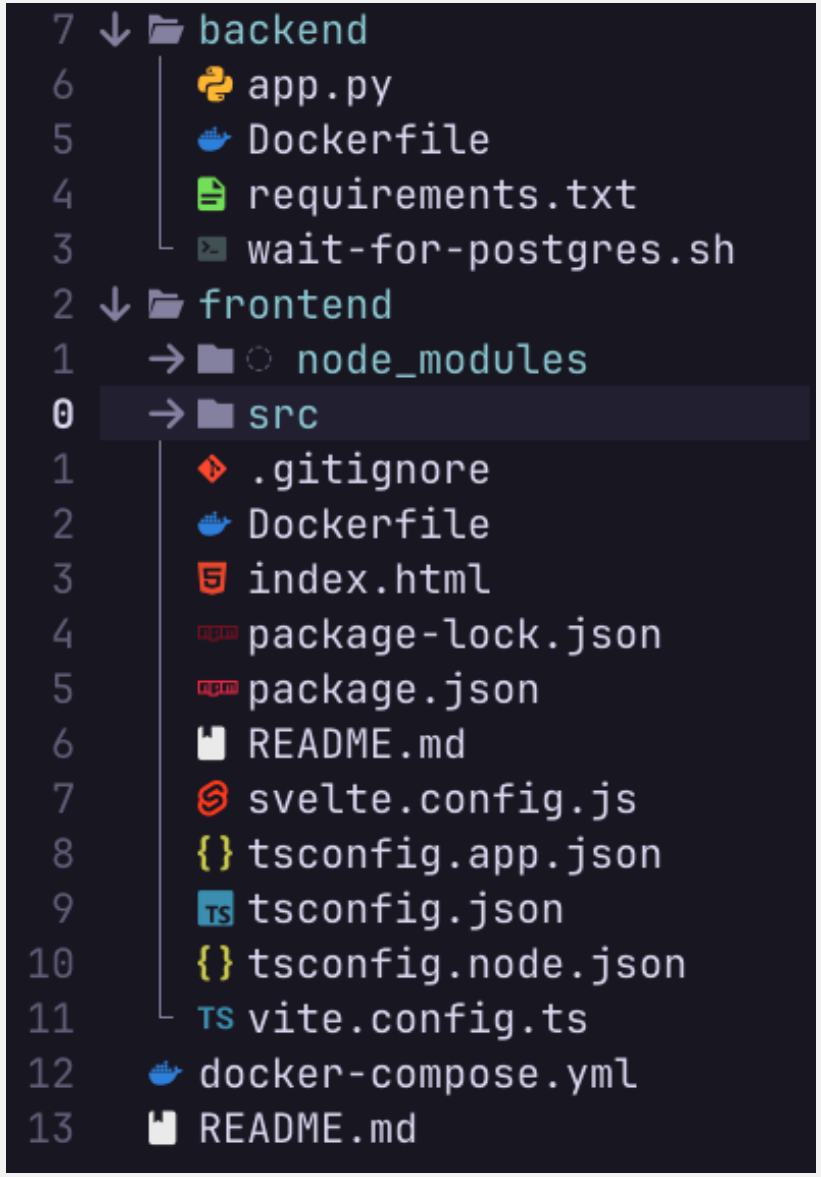

Real-Time Stock Tracker
Available Stocks
Enter stock symbol (e.g., AAPl
Add Stock
Connected to server: server-2
AAPL
Apple Inc.
$205.35
Server: server-2
AMZN
Amazon.com, Inc.
$189.98
Server: server-2

Architecture



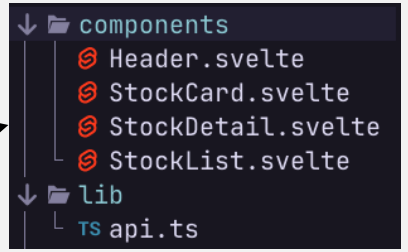
Monorepo

- Gut geeignet, da wenig Komplexität
- Einfache Versionierung
- Einfaches dependency management



Components

- Frontend
  - Real-time UI updates
  - Stock visualizing → Chart.js
  - Svelte-routing
  - Reusable Svelte Components
- Python
  - FastAPI
  - Database operations → Sqlalchemy
  - Yfinance (Yahoo! Finance API)
- PostgreSQL
  - Simpel zu benutzen
  - 2 tables (stracked_stocks & stock_prices)
  - Historie der stock Daten
- Traefik
  - Reverse Proxy
  - Round Robin load balancing (default)




```python
Base = declarative_base()
engine = create_async_engine(DATABASE_URL, echo=True, future=True)
async_session = sessionmaker(bind=engine, class_=AsyncSession, expire_on_commit=False)

class StockPrice(Base):
    __tablename__ = "stock_prices"

    id = Column(Integer, primary_key=True, index=True)
    symbol = Column(String, index=True)
    name = Column(String)
    price = Column(Float)
    timestamp = Column(DateTime, default=datetime.utcnow)
```

Design Decisions

- High availability
  - 2 backend Instanzen
  - Load balancing mit Traefik (default mit Round Robin)
  - Automatic failover
- Real-time Updates
  - SSE für daten-streaming
  - 5s update interval
- Monitoring und Maintenance
  - Health check endpoints
  - Traefik dashboard

Fazit

—

+ Einfache skalierung durch Docker + Traefik
+ Kein Testing
+ Spannendes Modul
+ Viel Freiheit
+ Challenge Task > Aufgaben

- Learnings
  - Traefik
  - ...

Schluss / Q&A