Phoenyra BESS Trade System - Dokumentation

Übersicht

Das Phoenyra BESS Trade System ist eine moderne Web-Anwendung für das Trading und die Optimierung von Battery Energy Storage Systems (BESS). Das System kombiniert eine FastAPI-basierte Backend-Architektur mit einer Flask-basierten Web-Oberfläche, die mit Tailwind CSS und Magic UI Komponenten gestaltet wurde.

Architektur

Backend-Services

- **Exchange Service** (FastAPI): Kern-Trading-Engine mit REST API
- **Redis**: In-Memory-Datenbank für Caching und Session-Management
- **Prometheus**: Metriken-Sammlung und Monitoring
- **Grafana**: Visualisierung und Alerting
- **Webapp Service** (Flask): Web-Dashboard und Benutzeroberfläche

Frontend-Technologien

- **Flask**: Python Web Framework
- **Tailwind CSS**: Utility-first CSS Framework
- **Magic UI**: Moderne UI-Komponenten und Effekte
- **ApexCharts**: Professionelle Chart-Bibliothek
- **Font Awesome**: Icon-Sammlung
- **Socket.IO**: Real-time Kommunikation

Features

1. Dashboard-Interface

- **Moderne Dark-Mode-Oberfläche** mit Gold-Standard-Design
- **Responsive Design** für Desktop und Mobile
- **Theme-Toggle** zwischen Dark und Light Mode
- **Live-Status-Anzeige** der Systemverbindung
- **Echtzeit-Uhr** in der Navigation

2. BESS-Status-Monitoring

- **State of Charge (SoC)** Anzeige in Prozent
- **Aktive Leistung** in MW
- **Temperatur-Monitoring** in °C
- **System-Status** (Online/Offline)
- **Echtzeit-Updates** alle 10 Sekunden

3. Trading-Funktionen

- **Order-Erstellung** mit Side (Buy/Sell), Menge, Preis und Markt

- **Aktive Orders** Übersicht
- **Recent Trades** Anzeige
- **Marktpreise** Visualisierung (Mark, EMA, VWAP)
- **VWAP-Chart** mit 15-Minuten-Intervallen

4. Chart-Visualisierung

- **ApexCharts Integration** für professionelle Charts
- **Marktpreise-Chart** mit Mark, EMA und VWAP
- **VWAP-Zeitreihen-Chart** mit 15-Minuten-Intervallen
- **Sattes Grün** für optimale Sichtbarkeit
- **Responsive Design** für verschiedene Bildschirmgrößen

5. Magic UI Komponenten

- **Aurora Text** für Überschriften
- **Neon Gradient Cards** für Status-Karten
- **Shimmer Buttons** für interaktive Elemente
- **Gold Standard Background** mit subtilen Gold-Akzenten
- **Hover-Effekte** und Animationen
- **Number Ticker** für animierte Zahlen

6. BESS Telemetrie-Steuerung

- **SoC-Management** mit Prozent-Eingabe
- **Leistungssteuerung** in MW
- **Temperatur-Überwachung** in °C
- **Telemetrie-Daten** an Backend senden

Technische Details

Docker-Container

```
services:
  exchange:     # FastAPI Backend (Port 9000)
  redis:        # In-Memory Database
  prometheus:   # Metrics Collection (Port 9090)
  grafana:      # Visualization (Port 3000)
  webapp:       # Flask Frontend (Port 5000)
```

API-Endpunkte

- `GET /api/bess-status` - BESS-Status abrufen
- `GET /api/market-data` - Marktdaten abrufen
- `GET /api/orders` - Aktive Orders abrufen
- `GET /api/trades` - Recent Trades abrufen
- `POST /api/orders` - Neue Order erstellen
- `POST /api/telemetry` - BESS-Telemetrie senden

Umgebungsvariablen

```
EXCHANGE_BASE_URL=http://exchange:9000
API_KEY=demo
HMAC_SECRET=phoenyra_demo_secret
FLASK_ENV=development
FLASK_DEBUG=1
```

Installation und Start

Voraussetzungen

- Docker und Docker Compose
- Python 3.11+
- Node.js (für WebSocket-Verbindungen)

Installation

```
# Repository klonen
git clone <repository-url>
cd phoenyra_BESS_Trade

# Docker Container starten
docker compose up -d --build

# Services überprüfen
docker compose ps
```

Zugriff auf Services

- **Web-Dashboard**: http://localhost:5000
- **Exchange API**: http://localhost:9000
- **Grafana**: http://localhost:3000
- **Prometheus**: http://localhost:9090

Entwicklung

Live-Reload

Das System unterstützt Live-Reload für Entwicklung:

- **Flask Debug Mode** aktiviert
- **Docker Volume Mounting** für sofortige Änderungen
- **Automatische Neuladung** bei Code-Änderungen

Code-Struktur

```
webapp/
├── app.py              # Flask Hauptanwendung
├── templates/
│   ├── base.html       # Basis-Template
│   └── dashboard.html  # Dashboard-Template
├── static/
│   └── phoenyra_logo.png  # Logo
└── Dockerfile          # Container-Definition
```

Design-System

Farbpalette

- **Primär**: Dunkle Grautöne (#1a1a1a, #2d2d2d)
- **Akzent**: Gold (#FFD700) für Highlights
- **Charts**: Sattes Grün (#00FF00) für optimale Sichtbarkeit
- **Text**: Weiß (#FFFFFF) für Kontrast

Magic UI Effekte

- **Gradient Backgrounds** mit subtilen Gold-Akzenten
- **Backdrop Blur** für moderne Glasmorphismus-Effekte
- **Hover-Animationen** für interaktive Elemente
- **Aurora Text** für dynamische Überschriften
- **Shimmer Buttons** für Call-to-Action-Elemente

Monitoring und Metriken

Prometheus Metriken

- **BESS-Status**: SoC, Leistung, Temperatur
- **Trading-Metriken**: Orders, Trades, Volumen
- **System-Performance**: API-Response-Zeiten, Fehlerraten

Grafana Dashboards

- **BESS-Überwachung**: SoC, Leistung, Temperatur-Trends
- **Trading-Analytics**: Order-Volumen, PnL, Marktpreise
- **System-Health**: API-Status, Response-Zeiten, Fehlerraten

Sicherheit

Authentifizierung

- **API-Key-basierte** Authentifizierung
- **HMAC-SHA256** Signaturen für WebSocket-Verbindungen
- **Key-Rotation** für erweiterte Sicherheit

Datenübertragung

- **HTTPS** für sichere Kommunikation
- **WebSocket-Verschlüsselung** für Real-time-Daten
- **Input-Validierung** für alle Benutzereingaben

Erweiterte Funktionen

Real-time Updates

- **WebSocket-Verbindungen** für Live-Daten
- **Automatische Aktualisierung** alle 10 Sekunden
- **Echtzeit-Chart-Updates** ohne Seitenneuladung

Responsive Design

- **Mobile-First** Ansatz
- **Flexible Grid-Layouts** für verschiedene Bildschirmgrößen
- **Touch-optimierte** Bedienelemente

Performance-Optimierung

- **Lazy Loading** für Chart-Komponenten
- **Efficient Data Fetching** mit Caching
- **Minimierte Bundle-Größe** für schnelle Ladezeiten

Troubleshooting

Häufige Probleme

1. **Container startet nicht**: Docker-Logs überprüfen
2. **Charts werden nicht angezeigt**: Browser-Konsole auf Fehler prüfen
3. **API-Verbindung fehlschlägt**: Exchange-Service-Status überprüfen

Debug-Modi

```
# Flask Debug-Modus
FLASK_DEBUG=1

# Docker-Logs anzeigen
docker compose logs -f webapp

# Container-Shell öffnen
docker exec -it webapp bash
```

Roadmap

Geplante Features

- **WebSocket-Integration** für Real-time-Updates
- **Erweiterte Chart-Analyse** mit technischen Indikatoren

- **Mobile App** für BESS-Monitoring
- **Machine Learning** für Preisvorhersagen
- **Multi-Market-Support** für verschiedene Energiebörsen

Performance-Verbesserungen

- **Caching-Strategien** für bessere Performance
- **Database-Optimierung** für große Datenmengen
- **CDN-Integration** für statische Assets

Support und Wartung

Logs und Monitoring

- **Strukturierte Logs** für einfache Fehlerdiagnose
- **Health-Checks** für alle Services
- **Automatische Alerts** bei kritischen Problemen

Updates und Wartung

- **Regelmäßige Security-Updates**
- **Performance-Monitoring** und Optimierung
- **Feature-Updates** basierend auf Benutzerfeedback

Diese Dokumentation beschreibt das Phoenyra BESS Trade System v2.0 mit allen implementierten Features und Funktionen.