TradeWatch UML Architecture Documentation

System Overview
TradeWatch Global Trade Intelligence Platform - Comprehensive UML Architecture and Component Design

---

System Architecture Diagram

Architecture Overview
The TradeWatch platform follows a layered architecture pattern with clear separation of concerns:

1. **Frontend Layer**: React-based user interface with interactive mapping
2. **API Gateway**: FastAPI server providing RESTful endpoints
3. **Data Processing Layer**: Real-time data fetching and validation services
4. **Database Layer**: PostgreSQL for persistent data storage
5. **AI/ML Pipeline**: TensorFlow-based prediction and analytics
6. **External Data Sources**: 15+ RSS feeds, government APIs, weather services

Data Flow
1. External data sources feed into the data processing layer
2. Real-time fetchers aggregate and validate incoming data
3. Processed data is stored in PostgreSQL database
4. AI/ML pipeline analyzes data for predictions
5. FastAPI serves data to React frontend
6. Interactive visualizations display real-time intelligence

Class Diagram

Component Relationships

Frontend Components
- **Dashboard**: Main application controller
- **GlobalMap**: Interactive Leaflet.js mapping component
- **VesselTracking**: Dedicated vessel monitoring interface

Backend Services
- **FastAPIServer**: Main API gateway and endpoint controller
- **RealTimeDisruptionFetcher**: RSS and news feed processing
- **RealAISIntegration**: Vessel tracking and AIS data management
- **PostgreSQLDatabase**: Data persistence and query layer

AI/ML Components
- **TensorFlowAI**: Machine learning models and prediction engine
- **CoordinateValidator**: Geospatial validation and land detection
- **DataAggregator**: Multi-source data fusion and quality control

Data Models
- **Vessel**: Maritime vessel data structure
- **Disruption**: Trade disruption and incident model
- **Port**: Global port operations and capacity data
- **Tariff**: International trade policy and regulation model

---

Technical Specifications

Frontend Architecture
```typescript
// React Component Hierarchy
Dashboard
â”œâ”€â”€ GlobalMap (Leaflet.js)
â”‚   â”œâ”€â”€ PortMarkers (200+ global ports)
â”‚   â”œâ”€â”€ DisruptionMarkers (122+ incidents)
â”‚   â”œâ”€â”€ VesselMarkers (5000+ vessels)
â”‚   â””â”€â”€ TariffOverlays (500+ policies)
â”œâ”€â”€ VesselTracking
```

```
â”‚   â”œâ”€â”€ FilterControls (country/route filtering)
â”‚   â”œâ”€â”€ VesselMap (dedicated vessel display)
â”‚   â””â”€â”€ VesselList (impacted vessel tracking)
â””â”€â”€ Analytics
    â”œâ”€â”€ AIProjections (TensorFlow predictions)
    â”œâ”€â”€ TrendAnalysis (historical patterns)
    â””â”€â”€ RiskAssessment (confidence scoring)
```

Backend Architecture
```python
# FastAPI Service Layer
FastAPIServer
â”œâ”€â”€ MaritimeDisruptionAPI (122+ real-time incidents)
â”œâ”€â”€ VesselTrackingAPI (5000+ AIS positions)
â”œâ”€â”€ PortInformationAPI (200+ major ports)
â”œâ”€â”€ TariffMonitoringAPI (500+ trade policies)
â””â”€â”€ AIPredictionAPI (80%+ confidence predictions)

# Data Processing Pipeline
DataProcessingLayer
â”œâ”€â”€ RealTimeDisruptionFetcher
â”‚   â”œâ”€â”€ RSSFeedParser (15+ maritime sources)
â”‚   â”œâ”€â”€ GovernmentAPIIntegrator (WTO, USTR, EU)
â”‚   â””â”€â”€ WeatherServiceIntegrator (NOAA, Weather Channel)
â”œâ”€â”€ AISIntegrationService
â”‚   â”œâ”€â”€ VesselPositionValidator (ocean-only filtering)
â”‚   â”œâ”€â”€ RouteOptimizationEngine (shipping lane analysis)
â”‚   â””â”€â”€ MaritimeCorridorValidator (geospatial verification)
â””â”€â”€ CoordinateValidator
    â”œâ”€â”€ LandDetectionEngine (advanced algorithms)
    â”œâ”€â”€ MaritimeRouteEngine (shipping lane validation)
    â””â”€â”€ ProximityAnalyzer (impact assessment)
```

Database Schema
```sql
-- Core Data Tables
PostgreSQL Database
â”œâ”€â”€ vessels_table (5000+ records)
â”‚   â”œâ”€â”€ id, imo, mmsi, name, type
â”‚   â”œâ”€â”€ coordinates, course, speed
â”‚   â”œâ”€â”€ origin, destination, flag
â”‚   â””â”€â”€ status, last_updated, impacted
â”œâ”€â”€ disruptions_table (122+ records)
â”‚   â”œâ”€â”€ id, title, description, type, severity
â”‚   â”œâ”€â”€ coordinates, affected_regions
â”‚   â”œâ”€â”€ start_date, end_date, confidence
â”‚   â””â”€â”€ event_type, sources, predictions
â”œâ”€â”€ ports_table (200+ records)
â”‚   â”œâ”€â”€ id, name, country, coordinates
â”‚   â”œâ”€â”€ strategic_importance, annual_teu
â”‚   â”œâ”€â”€ capacity_utilization, depth_meters
â”‚   â””â”€â”€ berths, crane_count, connectivity
â”œâ”€â”€ tariffs_table (500+ records)
â”‚   â”œâ”€â”€ id, name, type, rate, status
â”‚   â”œâ”€â”€ countries, products, effective_date
â”‚   â”œâ”€â”€ economic_impact, trade_volume
â”‚   â””â”€â”€ wto_case, sources, documentation
â””â”€â”€ ai_training_data (historical patterns)
    â”œâ”€â”€ feature_vectors, prediction_targets
    â”œâ”€â”€ confidence_scores, validation_results
    â””â”€â”€ model_performance_metrics
```

AI/ML Architecture
```python
# TensorFlow Model Pipeline
AIMLPipeline
â”œâ”€â”€ DataIngestion
â”‚   â”œâ”€â”€ HistoricalDataProcessor (5+ years)
â”‚   â”œâ”€â”€ RealTimeDataStreamer (30-second intervals)
â”‚   â””â”€â”€ FeatureEngineering (pattern extraction)
â”œâ”€â”€ ModelTraining
```

```
â”‚   â”œâ”€â”€ LSTMModels (sequence prediction)
â”‚   â”œâ”€â”€ CNNModels (pattern recognition)
â”‚   â””â”€â”€ EnsembleMethods (confidence aggregation)
â”œâ”€â”€ PredictionEngine
â”‚   â”œâ”€â”€ DisruptionForecasting (impact analysis)
â”‚   â”œâ”€â”€ VesselDelayPrediction (ETA optimization)
â”‚   â””â”€â”€ PortCongestionModeling (capacity analysis)
â””â”€â”€ ConfidenceScoring
    â”œâ”€â”€ SourceReliabilityWeighting (multi-factor)
    â”œâ”€â”€ TemporalConsistencyChecking (trend validation)
    â””â”€â”€ CrossValidationScoring (80%+ threshold)
```

Integration Patterns

Real-time Data Flow
```
External Sources â†’ Data Processing â†’ Database Storage â†’ AI Analysis â†’ API Serving â†’ Frontend Display
     â†“                  â†“             â†“              â†“             â†“            â†“
15+ RSS Feeds      Validation &    PostgreSQL     TensorFlow    FastAPI      Interactive
Government APIs    Aggregation     Real-time       Models        RESTful      Visualizations
Weather Services   Quality         Synchronization Predictions   Endpoints    & Analytics
News Sources       Filtering       ACID Compliance 80%+ Confidence Sub-200ms  Mobile Ready
```

Component Communication
- **Frontend â†” API**: RESTful HTTP requests with JSON payloads
- **API â†” Database**: PostgreSQL connections with connection pooling
- **API â†” AI/ML**: Direct Python function calls within FastAPI server
- **Data Processing â†” External**: HTTP/HTTPS requests with retry logic
- **AI/ML â†” Database**: SQL queries for training data and result storage

---

Performance Characteristics

System Metrics
- **API Response Time**: <200ms average
- **Database Query Performance**: Optimized with indexing
- **Real-time Update Frequency**: 30-second intervals
- **System Uptime**: 98.9% reliability target
- **Concurrent Users**: Scalable to 1000+ simultaneous

Data Capacity
- **Vessels Tracked**: 5000+ with real-time positioning
- **Disruptions Monitored**: 122+ active incidents
- **Ports Covered**: 200+ major global terminals
- **Tariffs Tracked**: 500+ international policies
- **Geographic Coverage**: Global maritime operations

Quality Assurance
- **Coordinate Accuracy**: Â±100m for vessel positions
- **Source Verification**: Multi-feed cross-reference
- **Prediction Confidence**: 80%+ minimum threshold
- **Data Freshness**: Real-time with 30-second updates

Deployment Architecture

Development Environment
```bash
Frontend: React + Vite development server (Port 5173)
Backend: FastAPI + Uvicorn ASGI server (Port 8001)
Database: PostgreSQL with real-time connections
AI/ML: TensorFlow with local GPU acceleration
```

Production Environment
```bash
Frontend: Nginx reverse proxy + optimized React build
Backend: Gunicorn + FastAPI with multiple workers
```

```
Database: PostgreSQL with read replicas + connection pooling
AI/ML: TensorFlow Serving with GPU clusters
Monitoring: Prometheus + Grafana + ELK stack
```

Security Architecture

Data Protection
- **Input Validation**: Comprehensive sanitization
- **CORS Security**: Controlled cross-origin access
- **Rate Limiting**: API abuse prevention
- **Encryption**: TLS 1.3 for data transmission

Authentication & Authorization
- **API Keys**: Service-to-service authentication
- **JWT Tokens**: User session management
- **Role-based Access**: Granular permission control
- **Audit Logging**: Comprehensive activity tracking

TradeWatch UML Architecture v2.1.0
Generated: January 2025
VectorStream Systems