



**UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS**

AI-POWERED PREDICTIVE TRADING ANALYTICS PLATFORM

David Alexander Colorado Rodríguez 20211020031
Andres Felipe Martín Rodriguez 20201020137

Data Bases II
Carlos Andres Sierra Virgüez

Universidad Distrital Francisco José de Caldas
Systems Engineering Curriculum Project
Colombia, Bogotá D.C.
May 13, 2025

Contents

1	Business Model Canvas	2
2	User Stories	2
2.1	Use Stories – Individual Trader	2
2.2	Use Stories – Portfolio Manager	3
2.3	Use Stories – Advanced Trader	4
3	Data Sources and Structures	4
4	Expected Data Volume and Transactions	5
5	Potential Problems and Considerations	5
6	Database Design	6
6.1	Define Components	6
6.1.1	Domain Layer (Core)	6
6.1.2	Ports (Interfaces)	6
6.1.3	Adapters	6
6.2	Define Entities	7
6.2.1	User Domain Entities	7
6.2.2	Trading Domain Entities	7
6.2.3	Market Data Entities	7
6.2.4	Portfolio Domain Entities	7
6.2.5	Analysis Domain Entities	7
6.3	Define Attributes per Entity	7
6.3.1	User Domain Attributes	8
6.3.2	Trading Domain Attributes	9
6.3.3	Market Data Attributes	11
6.3.4	Portfolio Domain Attributes	13
6.3.5	Analysis Domain Attributes	14
6.4	Define Relationship	16
6.5	Define Relationship Types	16
6.5.1	One-to-One (1:1) Relationships	16
6.5.2	One-to-Many (1:N) Relationships	16
6.5.3	Many-to-Many (M:N) Relationships	18
6.5.4	Reflexive Relationships	18
7	Entity-Relationship Diagram	19
8	References	19

1 Business Model Canvas

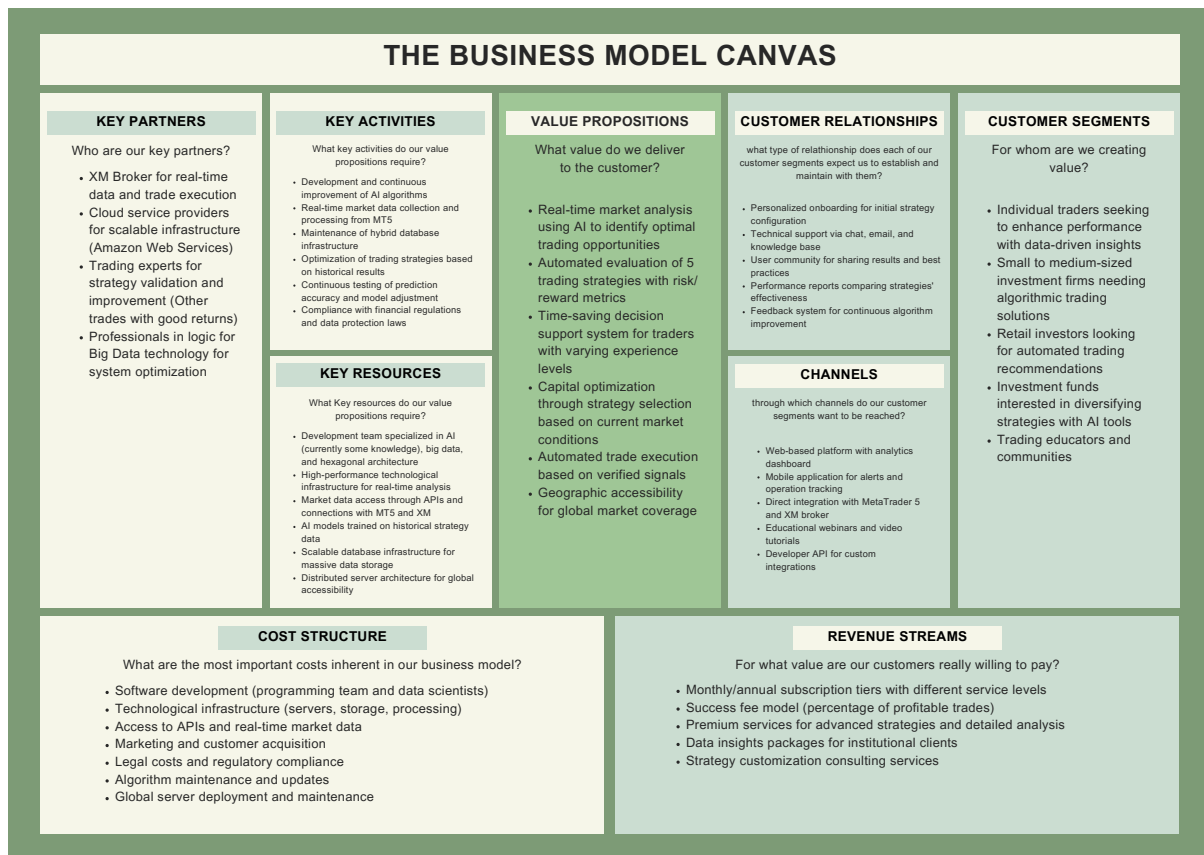


Figure 1: Business Model

2 User Stories

2.1 Use Stories – Individual Trader

Title: Real-Time Trading Alerts	Priority: High	Estimate: –
User Story: As an individual trader, I want to receive real-time alerts about buying/selling opportunities based on the analysis of the 5 strategies so that I can act quickly on market movements.		
Acceptance Criteria: Given the strategy is enabled When the system detects a market signal Then an alert is instantly delivered to the user		

Title: Mobile Platform Access	Priority: High	Estimate: –
User Story: As an individual trader, I want to access the trading platform from my mobile device so that I can stay updated and execute trades on the go.		
Acceptance Criteria: Given I have internet access When I log in from mobile Then I can use all trading features		

Title: Trade History Review	Priority: Medium	Estimate: –
User Story: As an individual trader, I want to review my past trades and strategy performance so that I can improve my future decisions.		
Acceptance Criteria: Given I'm logged in When I navigate to my trade history Then I can filter and view past operations		

2.2 Use Stories – Portfolio Manager

Title: Daily Strategy Performance Reports	Priority: High	Estimate: –
User Story: As a portfolio manager, I want to receive daily reports on each strategy's performance under various market conditions so that I can optimize capital distribution.		
Acceptance Criteria: Given I'm subscribed to reports When the day ends Then I receive a detailed summary of strategy results		

Title: Capital Allocation Simulation	Priority: High	Estimate: –
User Story: As a portfolio manager, I want to simulate how allocating capital across different strategies would have performed in the past so that I can make data-driven decisions.		
Acceptance Criteria: Given I input allocation percentages When I run the simulation Then the system displays historical performance results		

Title: Risk/Reward Monitoring	Priority: Medium-High	Estimate: –
User Story: As a portfolio manager, I want to monitor the risk/reward ratio of my strategy mix in real time so that I can proactively rebalance the portfolio.		
Acceptance Criteria: Given my portfolio is active When market data changes Then the system recalculates and alerts on deviations		

2.3 Use Stories – Advanced Trader

Title: AI Model Configuration	Priority: High	Estimate: –
User Story: As an advanced trader, I want to fine-tune AI model parameters so that I can adapt strategy behavior to my trading approach.		
Acceptance Criteria: Given I access the configuration panel When I change a model parameter Then recommendations are adjusted accordingly		

Title: Strategy Backtesting	Priority: High	Estimate: –
User Story: As an advanced trader, I want to run backtests with custom parameters so that I can validate my theories before trading live.		
Acceptance Criteria: Given I select a strategy and timeframe When I run a backtest Then the system shows detailed historical performance		

3 Data Sources and Structures

To obtain the raw financial data, given their ease of integration, usability in the market and wide recognition, the Yahoo Finance and Alpha Vantage sources were selected.

Yahoo Finance (yfinance)

Overview: yfinance is a popular library that provides access to Yahoo Finance data, including stock prices, financial statements, and historical data. Some features are:

- Retrieve historical market data
- Fetch fundamental data (income statements, balance sheets, cash flows)
- Get real-time stock prices
- Download options data

Alpha Vantage

Overview: Alpha Vantage provides free access to real-time and historical financial market data via APIs. Some features are:

- Stock, forex, and cryptocurrency data
- Fundamental data
- Technical indicators
- Economic indicators

Structures

The main structures that potentially will be needed are:

Customer data

User ID, name, email, strategy preferences, subscription level, transaction history.

Format: Relational database (e.g., PostgreSQL).

Transaction data

Trade ID, timestamp, strategy used, financial instrument, volume, result (profit/loss).

Format: Relational database with indexes for quick queries.

Real-time market data

Stock prices, forex, cryptocurrencies; technical indicators; volume.

Format: non-relational database (e.g., MongoDB) to handle massive flows.

Analytical data

Historical performance of strategies, risk/reward metrics, benchmark comparisons.

Format: data warehouse (e.g., Snowflake) for aggregated analysis.

Configuration data

User-customized parameters for AI models.

Format: Relational database.

4 Expected Data Volume and Transactions

Assumptions were used based on the customer segments and their needs as described in the user stories.

Users: Assume 10,000 individual traders, 500 investment firms and 1,000 retail investors in the first year (11,500 total users).

Interactions per user: Individual traders: 10 daily alerts, 5 trades executed.

Firms: 50 daily alerts, 20 trades.

Retail investors: 5 daily alerts, 2 trades.

Data generated per trade: 1 KB (includes timestamp, instrument, volume, result).

Daily volume:

Alerts: $(10,000 \times 10) + (500 \times 50) + (1,000 \times 5) = 130,000$ alerts.

Trades: $(10,000 \times 5) + (500 \times 20) + (1,000 \times 2) = 62,000$ trades.

Total data: $62,000 \times 1 \text{ KB} = 62 \text{ MB/day}$ (trades only) + market data (1 GB/day per tracked exchange).

Monthly volume: 1.86 GB (transactions) + 30 GB (market) = 32 GB/month (minimum).

Daily transactions: 62,000 trades executed.

5 Potential Problems and Considerations

Real-time data processing

Problem: Constant data ingestion from multiple exchanges (NYSE, NASDAQ, etc.) can overload infrastructure if not optimized.

Consideration: Use message queuing (e.g., Kafka) and distributed processing (e.g., Spark).

Scalability

Problem: User and data growth can degrade performance.

Consideration: Implement geographic replication and auto-scaling in AWS.

Security

Problem: Sensitive data (transactions, configurations) are targets for attacks.

Consideration: AES-256 encryption, multi-factor authentication and regular audits.

Regulatory compliance

Problem: Financial regulations vary by country (e.g., GDPR, SEC).

Consideration: Store data by region and generate automated compliance reports.

6 Database Design

6.1 Define Components

The components in a hexagonal architecture separate the core domain from external concerns. For your trading platform, these components are:

6.1.1 Domain Layer (Core)

Trading Domain

- Strategy evaluation logic
- Signal generation algorithms
- Risk calculation engines
- Portfolio optimization models

User Domain

- Authentication business rules
- Permission management
- Subscription handling logic

6.1.2 Ports (Interfaces)

Primary Ports (API Interfaces)

- StrategyService: Interface for strategy operations
- TradeService: Interface for trade execution
- AlertService: Interface for notification generation
- PortfolioService: Interface for portfolio management
- UserService: Interface for user operations

Secondary Ports (Repository Interfaces)

- StrategyRepository: Interface for strategy persistence
- TradeRepository: Interface for trade persistence
- MarketDataRepository: Interface for market data retrieval
- UserRepository: Interface for user persistence
- AlertRepository: Interface for alert persistence

6.1.3 Adapters

Primary Adapters (Driving)

- RESTController: REST API endpoints
- WebSocketController: Real-time communication
- ScheduledTaskRunner: Time-based operations

Secondary Adapters (Driven)

- **PostgreSQLRepository**: Implementation for relational data
- **MongoDBRepository**: Implementation for high-volume data
- **YahooFinanceAdapter**: Market data integration
- **AlphaVantageAdapter**: Additional market data
- **XMBrokerAdapter**: Trade execution with XM

6.2 Define Entities

6.2.1 User Domain Entities

1. **User**: System users with different roles (individual traders, portfolio managers, advanced traders)
2. **Subscription**: User subscription plans and payment status
3. **Role**: User permission roles (could be embedded in User or separate)
4. **UserPreference**: User settings and UI preferences

6.2.2 Trading Domain Entities

1. **Strategy**: Trading strategies with their logic and parameters
2. **StrategyConfig**: User-specific strategy configurations
3. **Trade**: Trading decisions and outcomes
4. **TradeExecution**: Broker interactions for trade execution
5. **Alert**: Trading opportunity notifications

6.2.3 Market Data Entities

1. **FinancialInstrument**: Tradable assets across markets
2. **MarketData**: Price and volume information
3. **TechnicalIndicator**: Calculated indicators based on market data

6.2.4 Portfolio Domain Entities

1. **Portfolio**: Collection of strategies with allocated capital
2. **PortfolioAllocation**: Distribution of capital across strategies
3. **PortfolioPerformance**: Performance metrics for portfolios

6.2.5 Analysis Domain Entities

1. **StrategyPerformance**: Performance metrics for strategies
2. **BacktestResult**: Results of historical strategy testing
3. **MarketCondition**: Classification of market environments

6.3 Define Attributes per Entity

Let me detail the specific attributes for each entity, including data types and constraints:

6.3.1 User Domain Attributes

User

- **user_id**: UUID (PK)
- **username**: VARCHAR(50) NOT NULL UNIQUE
- **email**: VARCHAR(100) NOT NULL UNIQUE
- **password_hash**: VARCHAR(255) NOT NULL
- **full_name**: VARCHAR(100) NOT NULL
- **created_at**: TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
- **last_login**: TIMESTAMP
- **user_type**: VARCHAR(20) NOT NULL CHECK (user_type IN ('individual', 'portfolio_manager', 'advanced'))
- **account_status**: VARCHAR(20) NOT NULL DEFAULT 'active'
- **verification_status**: BOOLEAN NOT NULL DEFAULT FALSE
- **profile_picture_url**: VARCHAR(255)
- **two_factor_enabled**: BOOLEAN NOT NULL DEFAULT FALSE

Subscription

- **subscription_id**: UUID (PK)
- **user_id**: UUID (FK to User)
- **subscription_level**: VARCHAR(20) NOT NULL CHECK (subscription_level IN ('free', 'basic', 'premium', 'enterprise'))
- **start_date**: TIMESTAMP NOT NULL
- **end_date**: TIMESTAMP NOT NULL
- **monthly_fee**: DECIMAL(10,2) NOT NULL
- **is_active**: BOOLEAN NOT NULL DEFAULT TRUE
- **auto_renew**: BOOLEAN NOT NULL DEFAULT TRUE
- **payment_method_id**: VARCHAR(100)
- **last_payment_date**: TIMESTAMP
- **next_payment_date**: TIMESTAMP

Role (if separate from User)

- **role_id**: UUID (PK)
- **name**: VARCHAR(50) NOT NULL UNIQUE
- **description**: TEXT
- **permissions**: JSONB

UserPreference

- **preference_id**: UUID (PK)
- **user_id**: UUID (FK to User)
- **theme**: VARCHAR(20) DEFAULT 'light'
- **notification_settings**: JSONB
- **default_timeframe**: VARCHAR(10) DEFAULT '1d'
- **default_indicators**: JSONB
- **ui_layout**: JSONB
- **alert_preferences**: JSONB

6.3.2 Trading Domain Attributes

Strategy

- **strategy_id**: UUID (PK)
- **name**: VARCHAR(100) NOT NULL
- **description**: TEXT
- **type**: VARCHAR(50) NOT NULL
- **created_at**: TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
- **is_active**: BOOLEAN NOT NULL DEFAULT TRUE
- **default_parameters**: JSONB NOT NULL
- **version**: VARCHAR(20) NOT NULL
- **creator**: VARCHAR(100)
- **performance_summary**: JSONB
- **risk_level**: VARCHAR(20)

StrategyConfig

- **config_id**: UUID (PK)
- **user_id**: UUID (FK to User)
- **strategy_id**: UUID (FK to Strategy)
- **parameters**: JSONB NOT NULL
- **created_at**: TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
- **updated_at**: TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
- **is_active**: BOOLEAN NOT NULL DEFAULT TRUE
- **name**: VARCHAR(100)
- **description**: TEXT
- **performance_summary**: JSONB
- **is_favorite**: BOOLEAN DEFAULT FALSE

Trade

- **trade_id**: UUID (PK)
- **user_id**: UUID (FK to User)
- **instrument_id**: UUID (FK to FinancialInstrument)
- **data_id**: UUID (FK to MarketData)
- **strategy_id**: UUID (FK to Strategy)
- **config_id**: UUID (FK to StrategyConfig)
- **direction**: VARCHAR(10) NOT NULL CHECK (direction IN ('buy', 'sell'))
- **volume**: DECIMAL(18,8) NOT NULL
- **entry_price**: DECIMAL(18,8) NOT NULL
- **exit_price**: DECIMAL(18,8)
- **stop_loss**: DECIMAL(18,8)
- **take_profit**: DECIMAL(18,8)
- **entry_time**: TIMESTAMP NOT NULL
- **exit_time**: TIMESTAMP
- **profit_loss**: DECIMAL(18,8)
- **profit_loss_percentage**: DECIMAL(10,4)
- **commission**: DECIMAL(18,8) NOT NULL DEFAULT 0
- **status**: VARCHAR(20) NOT NULL CHECK (status IN ('planned', 'active', 'completed', 'cancelled'))
- **notes**: TEXT
- **tags**: VARCHAR[]

TradeExecution

- **execution_id**: UUID (PK)
- **trade_id**: UUID (FK to Trade)
- **broker_reference**: VARCHAR(100)
- **execution_status**: VARCHAR(20) NOT NULL
- **execution_time**: TIMESTAMP NOT NULL
- **executed_price**: DECIMAL(18,8) NOT NULL
- **executed_volume**: DECIMAL(18,8) NOT NULL
- **execution_details**: JSONB
- **latency_ms**: INTEGER
- **broker_commission**: DECIMAL(18,8)
- **slippage**: DECIMAL(18,8)

Alert

- **alert_id:** UUID (PK)
- **user_id:** UUID (FK to User)
- **strategy_id:** UUID (FK to Strategy)
- **data_id:** UUID (FK to MarketData)
- **alert_type:** VARCHAR(50) NOT NULL
- **message:** TEXT NOT NULL
- **generated_at:** TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
- **sent_at:** TIMESTAMP
- **is_read:** BOOLEAN NOT NULL DEFAULT FALSE
- **confidence_score:** DECIMAL(5,2) NOT NULL
- **instrument_id:** UUID (FK to FinancialInstrument)
- **timeframe:** VARCHAR(10)
- **priority:** VARCHAR(10) DEFAULT 'medium'
- **expiration_time:** TIMESTAMP

6.3.3 Market Data Attributes

FinancialInstrument

- **instrument_id:** UUID (PK)
- **symbol:** VARCHAR(20) NOT NULL
- **name:** VARCHAR(100) NOT NULL
- **type:** VARCHAR(20) NOT NULL CHECK (type IN ('stock', 'forex', 'crypto', 'commodity', 'index', 'bond'))
- **exchange:** VARCHAR(50) NOT NULL
- **currency:** VARCHAR(10) NOT NULL
- **is_active:** BOOLEAN NOT NULL DEFAULT TRUE
- **description:** TEXT
- **sector:** VARCHAR(100)
- **country:** VARCHAR(100)
- **lot_size:** DECIMAL(18,8)
- **min_tick:** DECIMAL(18,8)
- **trading_hours:** JSONB
- **margin_requirements:** DECIMAL(5,2)

MarketData

- **data_id:** UUID (PK)
- **instrument_id:** UUID (FK to FinancialInstrument)
- **timestamp:** TIMESTAMP NOT NULL
- **open_price:** DECIMAL(18,8) NOT NULL
- **high_price:** DECIMAL(18,8) NOT NULL
- **low_price:** DECIMAL(18,8) NOT NULL
- **close_price:** DECIMAL(18,8) NOT NULL
- **volume:** DECIMAL(18,8) NOT NULL
- **timeframe:** VARCHAR(10) NOT NULL
- **data_source:** VARCHAR(50) NOT NULL
- **adjusted_close:** DECIMAL(18,8)
- **bid:** DECIMAL(18,8)
- **ask:** DECIMAL(18,8)
- **spread:** DECIMAL(18,8)
- **partition_key:** VARCHAR(50) NOT NULL

TechnicalIndicator

- **indicator_id:** UUID (PK)
- **data_id:** UUID (FK to MarketData)
- **indicator_type:** VARCHAR(50) NOT NULL
- **parameters:** JSONB NOT NULL
- **values:** JSONB NOT NULL
- **calculated_at:** TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
- **instrument_id:** UUID (FK to FinancialInstrument)
- **timeframe:** VARCHAR(10) NOT NULL
- **validity_period:** TIMESTAMP
- **signal_strength:** DECIMAL(5,2)
- **partition_key:** VARCHAR(50) NOT NULL

6.3.4 Portfolio Domain Attributes

Portfolio

- **portfolio_id:** UUID (PK)
- **user_id:** UUID (FK to User)
- **name:** VARCHAR(100) NOT NULL
- **description:** TEXT
- **initial_capital:** DECIMAL(18,2) NOT NULL
- **created_at:** TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
- **is_active:** BOOLEAN NOT NULL DEFAULT TRUE
- **currency:** VARCHAR(10) NOT NULL DEFAULT 'USD'
- **risk_profile:** VARCHAR(20)
- **target_return:** DECIMAL(5,2)
- **max_drawdown_limit:** DECIMAL(5,2)
- **rebalancing_frequency:** VARCHAR(20)

PortfolioAllocation

- **allocation_id:** UUID (PK)
- **portfolio_id:** UUID (FK to Portfolio)
- **strategy_id:** UUID (FK to Strategy)
- **config_id:** UUID (FK to StrategyConfig)
- **allocation_percentage:** DECIMAL(5,2) NOT NULL
- **updated_at:** TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
- **allocation_amount:** DECIMAL(18,2) NOT NULL
- **is_active:** BOOLEAN NOT NULL DEFAULT TRUE
- **notes:** TEXT
- **performance_contribution:** DECIMAL(5,2)

PortfolioPerformance

- **performance_id:** UUID (PK)
- **portfolio_id:** UUID (FK to Portfolio)
- **period_start:** TIMESTAMP NOT NULL
- **period_end:** TIMESTAMP NOT NULL
- **current_value:** DECIMAL(18,2) NOT NULL
- **profit_loss:** DECIMAL(18,2) NOT NULL
- **return_percentage:** DECIMAL(10,4) NOT NULL
- **sharpe_ratio:** DECIMAL(10,4)
- **max_drawdown:** DECIMAL(5,2) NOT NULL

- **volatility:** DECIMAL(10,4) NOT NULL
- **alpha:** DECIMAL(10,4)
- **beta:** DECIMAL(10,4)
- **calmar_ratio:** DECIMAL(10,4)
- **sortino_ratio:** DECIMAL(10,4)
- **market_correlation:** DECIMAL(5,2)

6.3.5 Analysis Domain Attributes

StrategyPerformance

- **performance_id:** UUID (PK)
- **strategy_id:** UUID (FK to Strategy)
- **config_id:** UUID (FK to StrategyConfig)
- **data_id:** UUID (FK to MarketData)
- **period_start:** TIMESTAMP NOT NULL
- **period_end:** TIMESTAMP NOT NULL
- **win_rate:** DECIMAL(5,2) NOT NULL
- **profit_factor:** DECIMAL(10,4) NOT NULL
- **max_drawdown:** DECIMAL(5,2) NOT NULL
- **sharpe_ratio:** DECIMAL(10,4) NOT NULL
- **total_trades:** INTEGER NOT NULL
- **winning_trades:** INTEGER NOT NULL
- **losing_trades:** INTEGER NOT NULL
- **avg_profit_loss:** DECIMAL(18,8) NOT NULL
- **avg_win:** DECIMAL(18,8)
- **avg_loss:** DECIMAL(18,8)
- **market_condition:** VARCHAR(20)
- **instrument_id:** UUID (FK to FinancialInstrument)
- **timeframe:** VARCHAR(10) NOT NULL

BacktestResult

- **backtest_id:** UUID (PK)
- **strategy_id:** UUID (FK to Strategy)
- **config_id:** UUID (FK to StrategyConfig)
- **period_start:** TIMESTAMP NOT NULL
- **period_end:** TIMESTAMP NOT NULL
- **initial_capital:** DECIMAL(18,2) NOT NULL
- **final_capital:** DECIMAL(18,2) NOT NULL

- **total_profit_loss:** DECIMAL(18,2) NOT NULL
- **win_rate:** DECIMAL(5,2) NOT NULL
- **max_drawdown:** DECIMAL(5,2) NOT NULL
- **sharpe_ratio:** DECIMAL(10,4) NOT NULL
- **total_trades:** INTEGER NOT NULL
- **created_at:** TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
- **detailed_results:** JSONB
- **equity_curve:** JSONB
- **monthly_returns:** JSONB
- **instruments_tested:** UUID[] (Array of FinancialInstrument IDs)
- **market_conditions:** VARCHAR[]

MarketCondition

- **condition_id:** UUID (PK)
- **name:** VARCHAR(50) NOT NULL
- **description:** TEXT
- **parameters:** JSONB
- **start_date:** TIMESTAMP
- **end_date:** TIMESTAMP
- **instrument_id:** UUID (FK to FinancialInstrument)
- **volatility_level:** DECIMAL(5,2)
- **trend_direction:** VARCHAR(20)
- **indicators_state:** JSONB

6.4 Define Relationship

	Market_Data	Financial_Instrument	Trade	Trade_Execution	User	Subscription	Alert	Strategy	Strategy_Config	Strategy_performance	Technical_Indicator	Backtest_Result	Portfolio	Portfolio_Allocation	Portfolio_Performance
Market_Data	X	X					X			X	X				
Financial_Instrument	X	X	X												
Trade	X	X	X	X	X										
Trade_Execution			X	X											
User			X		X	X	X		X				X		
Subscription					X	X									
Alert	X				X		X	X							
Strategy							X	X	X	X		X		X	
Strategy_Config					X			X	X						
Strategy_performance	X							X		X					
Technical_Indicator	X										X				
Backtest_Result								X				X			
Portfolio					X								X	X	X
Portfolio_Allocation													X	X	
Portfolio_Performance													X		X

Figure 2: Relaciones Modelo ER

6.5 Define Relationship Types

Here are the specific relationship types with cardinality constraints:

6.5.1 One-to-One (1:1) Relationships

label=0. **User - UserPreference**

- Each user has exactly one set of preferences.
- Implementation: Foreign key from **UserPreference** to **User** with unique constraint.

6.5.2 One-to-Many (1:N) Relationships

label=0. **User - Subscription**

- A user can have multiple subscriptions (historical record).
- Implementation: Foreign key from **Subscription** to **User**.

lbbel=0. **User - StrategyConfig**

- A user can create multiple strategy configurations.
- Implementation: Foreign key from **StrategyConfig** to **User**.

lcbel=0. **User - Trade**

- A user can execute multiple trades.
- Implementation: Foreign key from **Trade** to **User**.

ldbel=0. **User - Alert**

- A user can receive multiple alerts.

- Implementation: Foreign key from **Alert** to **User**.

lebel=0. **User - Portfolio**

- A user can manage multiple portfolios.
- Implementation: Foreign key from **Portfolio** to **User**.

lfbel=0. **Strategy - StrategyConfig**

- A strategy can have multiple configurations.
- Implementation: Foreign key from **StrategyConfig** to **Strategy**.

lgbel=0. **Strategy - Alert**

- A strategy can generate multiple alerts.
- Implementation: Foreign key from **Alert** to **Strategy**.

lhbel=0. **Strategy - StrategyPerformance**

- A strategy has multiple performance records.
- Implementation: Foreign key from **StrategyPerformance** to **Strategy**.

libel=0. **Strategy - BacktestResult**

- A strategy can have multiple backtest results.
- Implementation: Foreign key from **BacktestResult** to **Strategy**.

ljbel=0. **FinancialInstrument - MarketData**

- An instrument generates multiple market data points.
- Implementation: Foreign key from **MarketData** to **FinancialInstrument**.

lkbel=0. **FinancialInstrument - Trade**

- An instrument can be involved in multiple trades.
- Implementation: Foreign key from **Trade** to **FinancialInstrument**.

llbel=0. **MarketData - TechnicalIndicator**

- A market data point can generate multiple technical indicators.
- Implementation: Foreign key from **TechnicalIndicator** to **MarketData**.

lmbel=0. **Trade - TradeExecution**

- A trade can have multiple execution records.
- Implementation: Foreign key from **TradeExecution** to **Trade**.

lnbel=0. **Portfolio - PortfolioAllocation**

- A portfolio contains multiple strategy allocations.
- Implementation: Foreign key from **PortfolioAllocation** to **Portfolio**.

lobel=0. **Portfolio - PortfolioPerformance**

- A portfolio has multiple performance records.
- Implementation: Foreign key from **PortfolioPerformance** to **Portfolio**.

lpbel=0. **StrategyConfig - BacktestResult**

- A configuration can have multiple backtest results.
- Implementation: Foreign key from **BacktestResult** to **StrategyConfig**.

6.5.3 Many-to-Many (M:N) Relationships

label=0. **Strategy - Portfolio**

- A strategy can be used in multiple portfolios.
- A portfolio can use multiple strategies.
- Implementation: Resolved through **PortfolioAllocation** junction table.

lbbel=0. **Strategy - MarketCondition**

- A strategy can perform differently under multiple market conditions.
- A market condition affects multiple strategies.
- Implementation: Relationship is captured through **StrategyPerformance** table.

lcbel=0. **FinancialInstrument - Strategy**

- A strategy can trade multiple instruments.
- An instrument can be traded by multiple strategies.
- Implementation: Implicit through **Trade** records.

6.5.4 Reflexive Relationships

label=0. **Strategy - Strategy (Base/Derived)**

- A strategy can be derived from another strategy.
- Implementation: Add **parent_strategy_id** (nullable FK) to **Strategy** table.

lbbel=0. **Portfolio - Portfolio (Master/Sub)**

- A portfolio can contain sub-portfolios.
- Implementation: Add **parent_portfolio_id** (nullable FK) to **Portfolio** table.

7 Entity-Relationship Diagram

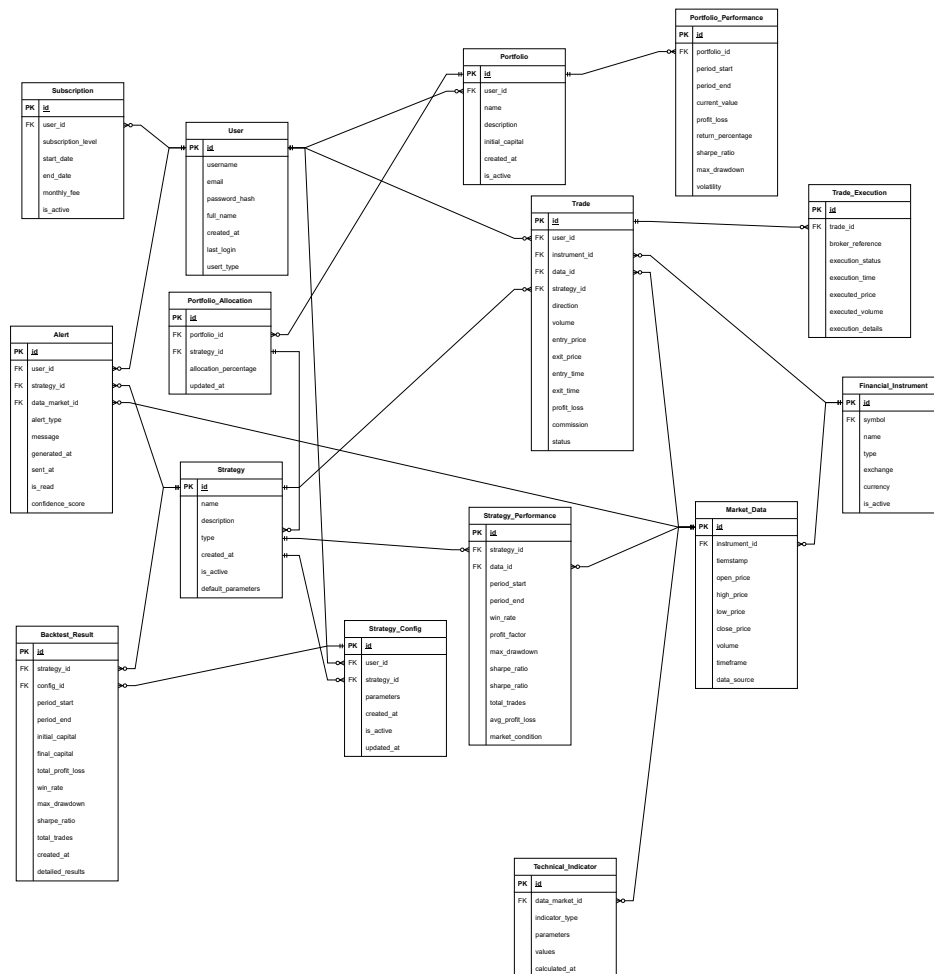


Figure 3: Model ER

Link al repositorio: <https://github.com/felimarod/project-databases-ii>

8 References