

Server Interview – Orders & Trades

This module handles **orders with nested trades** and **real-time updates** for a Forex trading system.

The **screenshot** illustrates how orders and trades should appear use it as the basis for your schema and structure design.

Forex Orders & Trades

Order ID: 64f9abc123

Pair: EUR/USD Type: **BUY** Amount: 1000 Price: 1.105

750 / 1000

View Trades

Trade ID: 64f9abc123h24g

Executed: 250

Price: 1.105

14:25

Trade ID: 64f9abc123h25g

Executed: 500

Price: 1.106

14:30

Order ID: 64f9abc124

Pair: USD/JPY Type: **SELL** Amount: 2000 Price: 145.20

2000 / 2000

View Trades

Trade ID: 64f9abc123h27g

Executed: 2000

Price: 145.20

15:15

Requirements

Orders

- Contain details such as **pair**, **type**, **amount**, **price**, **status**.
- Have a list of **trades** beneath them.
- Support status progression: **PENDING** → **PARTIALLY_FILLED** → **FILLED**

Trades

- Represent **individual executions** (amount, price, timestamp).
- Adding trades updates the order's **filled progress** and **status** automatically.

Business Rules

- The sum of all executed trades must not exceed the order's amount.
- If total filled **< amount** → **PARTIALLY_FILLED**.
- If total filled **= amount** → **FILLED**.
- If adding a trade would overshoot → reject (400).

API Endpoints (JWT required)

Method	Path	Purpose
POST	/orders	Create a new order
POST	/orders/:id/trades	Add a trade (updates order status automatically)

Simulation (Swagger/Postman)

- **Create an order** (e.g., BUY EUR/USD 1000).
 - **Add trades** (e.g., 250 + 500 executed).
 - Verify that the order status updates automatically (`PARTIALLY_FILLED` , then `FILLED`).
-

Deliverables

- Implement Orders module.
- Add API endpoints + validation.
- Test simulation with Swagger/Postman.