Place Order → Match → Record On-Chain (Sequence)

Participants: User | Frontend / UI | Order Service (API) | Database | Matching Engine | Notification Service / WebSocket | Blockchain Service | Smart Contract (Private Blockchain)

- User → Frontend / UI: placeOrder(symbol, side, price, qty)
- Frontend / UI → Order Service (API): POST /orders (JWT, payload)
- Order Service (API) → Order Service (API): validateOrder()

alt [invalid]
- Order Service (API) → Frontend / UI: 400 rejected(reason)

  opt [(optional)]
  - Order Service (API) → Notification Service / WebSocket: order.rejected(reason)

[valid]
- Order Service (API) → Database: insert Order(status=NEW)
- Database → Order Service (API): orderId
- Order Service (API) → Notification Service / WebSocket: order.created(orderId, NEW)
- Order Service (API) → Matching Engine: submit(orderId)
- Matching Engine → Matching Engine: match(orderBook)

  alt [no match]
  - Matching Engine → Database: update Order(status=NEW/QUEUED)
  - Matching Engine → Notification Service / WebSocket: order.updated(orderId, QUEUED)

  [match found]
  - Matching Engine → Database: create Trade + update Orders(PARTIAL/FILLED)
  - Matching Engine → Notification Service / WebSocket: trade.executed + order.updated
  - Matching Engine → Blockchain Service: recordTrade(tradeData)
  - Blockchain Service → Smart Contract (Private Blockchain): recordTrade(...)
  - Smart Contract (Private Blockchain) → Blockchain Service: txHash / receipt

    alt [tx failed]
    - Blockchain Service → Database: update Trade(status=PENDING_ONCHAIN, errorInfo)
    - Blockchain Service → Notification Service / WebSocket: trade.onchain_pending(tradeId)

    [tx confirmed]
    - Blockchain Service → Database: update Trade(txHash, status=RECORDED_ONCHAIN)
    - Blockchain Service → Notification Service / WebSocket: trade.recorded_onchain(txHash)

- Notification Service / WebSocket → Frontend / UI: push updates (order/trade events)
- Frontend / UI → User: show status