QS server ouptut data

Data is provided in publisher/subscriber pattern.

After subscribin with using REQ/RESP pattern Server can provide the following data:
OrderBook message with up-to-date asks and bids orders contining size, price and order id.
All messages definition are located under protos/qs_protos/Message.proto

On the client side you should use protos/qs_protos/Message.proto to implement deserialization. See client.cpp example

Trade messages

QS server provides also trade messages with additional information. Currentyl supported trade types:

```
enum TradeType {
    TradeType_ACTIVATE = 0;
    TradeType_CHANGE = 1;
    TradeType_DONE = 2;
    TradeType_MATCH = 3;
    TradeType_OPEN = 4;
}
```

All details can be found in the protos/qs_protos/Message.proto file:

```
syntax = "proto2";
package qs;

enum MessageType {
    MessageType_ORDERBOOK = 0;
    MessageType_TRADE = 1;
}

enum TradeType {
    TradeType_ACTIVATE = 0;
    TradeType_CHANGE = 1;
    TradeType_DONE = 2;
    TradeType_MATCH = 3;
    TradeType_OPEN = 4;
}

message Order {
  required string order_id = 1;
  required float price = 2;
  required float size = 3;
}

message OrderBook {
  required string product_id = 1;
  repeated Order asks = 2;
  repeated Order bids = 3;
}

message Trade {
  required TradeType type = 1;
  required string product_id = 2;
  required string order_id = 3;
  optional float size = 4;
  optional float price = 5;
  optional float founds = 6;
  optional float stop_price = 7;
  optional string stop_type = 8;
  optional string order_type = 9;
  optional float funds = 10;
}

message qsMessage{
    required MessageType message_type = 1;
    oneof msg {
        Trade trade = 2;
        OrderBook order_book = 3;
    }
}
```