DTS NodeJS Capstone Assignment

Name: Atharva Athanikar 72287317E

Batch: DTS Batch-3

Structure:

1. Stocks-trade-api
   * 1. Config
        + 1. db.cjs
     2. controllers
        + 1. tradeController.cjs
     3. middlewares
        + 1. authMiddleware.cjs
     4. models
        + 1. trade.cjs
     5. routes
        + 1. tradeRoutes.cjs
     6. tradeApi.cjs

./config/db.cjs

const mongoose = require("mongoose");

mongoose.connect("mongodb://localhost:27017/stocksTrade", {

  useNewUrlParser: true,

  useUnifiedTopology: true,

})

.then(() => console.log("Connected to MongoDB"))

.catch((err) => console.log("Error connecting to MongoDB:", err));

./controllers/tradeController.cjs

const Trade = require("../models/trade.cjs");

const createTrade = async (req, res) => {

  try {

    const { type, user\_id, symbol, shares, price, timestamp } = req.body;

    if (shares < 1 || shares > 100) {

      return res.status(400).json({ message: "Shares value out of range" });

    }

    if (type !== "buy" && type !== "sell") {

      return res.status(400).json({ message: "Incorrect type provided" });

    }

    const trade = new Trade({ type, user\_id, symbol, shares, price, timestamp });

    await trade.save();

    res.status(201).json(trade);

  } catch (error) {

    res.status(500).json({ message: error.message });

  }

};

const getAllTrades = async (req, res) => {

  try {

    const { type, user\_id } = req.query;

    let filters = {};

    if (type) filters.type = type;

    if (user\_id) filters.user\_id = user\_id;

    const trades = await Trade.find(filters);

    res.status(200).json(trades);

  } catch (error) {

    res.status(500).json({ message: error.message });

  }

};

const getTradeById = async (req, res) => {

  try {

    const trade = await Trade.findById(req.params.id);

    if (!trade) {

      return res.status(404).json({ message: "ID not found" });

    }

    res.status(200).json(trade);

  } catch (error) {

    res.status(500).json({ message: error.message });

  }

};

module.exports = { createTrade, getAllTrades, getTradeById };

./middlewares/authMiddleware.cjs

const jwt = require("jsonwebtoken");

const authMiddleware = (req, res, next) => {

  const token = req.header("Authorization")?.split(" ")[1];

  if (!token) return res.status(401).json({ message: "Access denied" });

  try {

    const decoded = jwt.verify(token, "yourSecretKey");

    req.user = decoded;

    next();

  } catch (err) {

    return res.status(400).json({ message: "Invalid token" });

  }

};

module.exports = authMiddleware;

./models/trade.cjs

const mongoose = require("mongoose");

const tradeSchema = new mongoose.Schema({

  type: { type: String, enum: ["buy", "sell"], required: true },

  user\_id: { type: Number, required: true },

  symbol: { type: String, required: true },

  shares: { type: Number, required: true, min: 1, max: 100 },

  price: { type: Number, required: true },

  timestamp: { type: Number, required: true },

});

module.exports = mongoose.model("Trade", tradeSchema);

./routes/tradeRoutes.cjs

const { Router } = require("express");

const router = Router();

const { createTrade, getAllTrades, getTradeById } = require("../controllers/tradeController.cjs");

const authMiddleware = require("../middlewares/authMiddleware.cjs");

// POST /trades - Create a new trade

router.post("/trades", authMiddleware, createTrade);

// GET /trades - Get all trades

router.get("/trades", authMiddleware, getAllTrades);

// GET /trades/:id - Get a trade by id

router.get("/trades/:id", authMiddleware, getTradeById);

// Catch-all for disallowed methods on /trades/:id

router.delete("/trades/:id", (req, res) => {

  res.status(405).json({ message: "Method not allowed" });

});

router.patch("/trades/:id", (req, res) => {

  res.status(405).json({ message: "Method not allowed" });

});

module.exports = router;

tradeApi.cjs / Main app

const express = require("express");

const { MongoClient } = require("mongodb");

const { v4: uuidv4 } = require("uuid");

const app = express();

app.use(express.json());

const mongoUrl = "mongodb://127.0.0.1:27017/Capstone-Project\_DTS";

const dbName = "stocksTrade";

let db, tradesCollection;

MongoClient.connect(mongoUrl, { useUnifiedTopology: true })

    .then((client) => {

        db = client.db(dbName);

        tradesCollection = db.collection("trades");

        console.log(`Connected to database: ${dbName}`);

    })

    .catch((error) => {

        console.error("Error connecting to MongoDB:", error);

        process.exit(1);

    });

const trades = [];

app.post("/trades", (req, res) => {

    const { type, user\_id, symbol, shares, price, timestamp } = req.body;

    if (!type || !user\_id || !symbol || !shares || !price || !timestamp) {

        return res.status(400).json({ message: "Missing required fields" });

    }

    if (type !== "buy" && type !== "sell") {

        return res.status(400).json({ message: "Incorrect type provided" });

    }

    if (shares < 1 || shares > 100) {

        return res.status(400).json({ message: "Shares value out of range" });

    }

    const newTrade = {

        id: uuidv4(),

        type,

        user\_id,

        symbol,

        shares,

        price,

        timestamp,

    };

    trades.push(newTrade);

    res.status(201).json(newTrade);

});

app.get("/trades", (req, res) => {

    const { type, user\_id } = req.query;

    let filteredTrades = trades;

    if (type) {

        filteredTrades = filteredTrades.filter((trade) => trade.type === type);

    }

    if (user\_id) {

        filteredTrades = filteredTrades.filter((trade) => trade.user\_id === parseInt(user\_id));

    }

    res.status(200).json(filteredTrades);

});

app.delete('/trades/:id', (req, res) => {

  res.status(405).json({ message: "Method Not Allowed" });

});

app.patch('/trades/:id', (req, res) => {

  res.status(405).json({ message: "Method Not Allowed" });

});

const PORT = 8000;

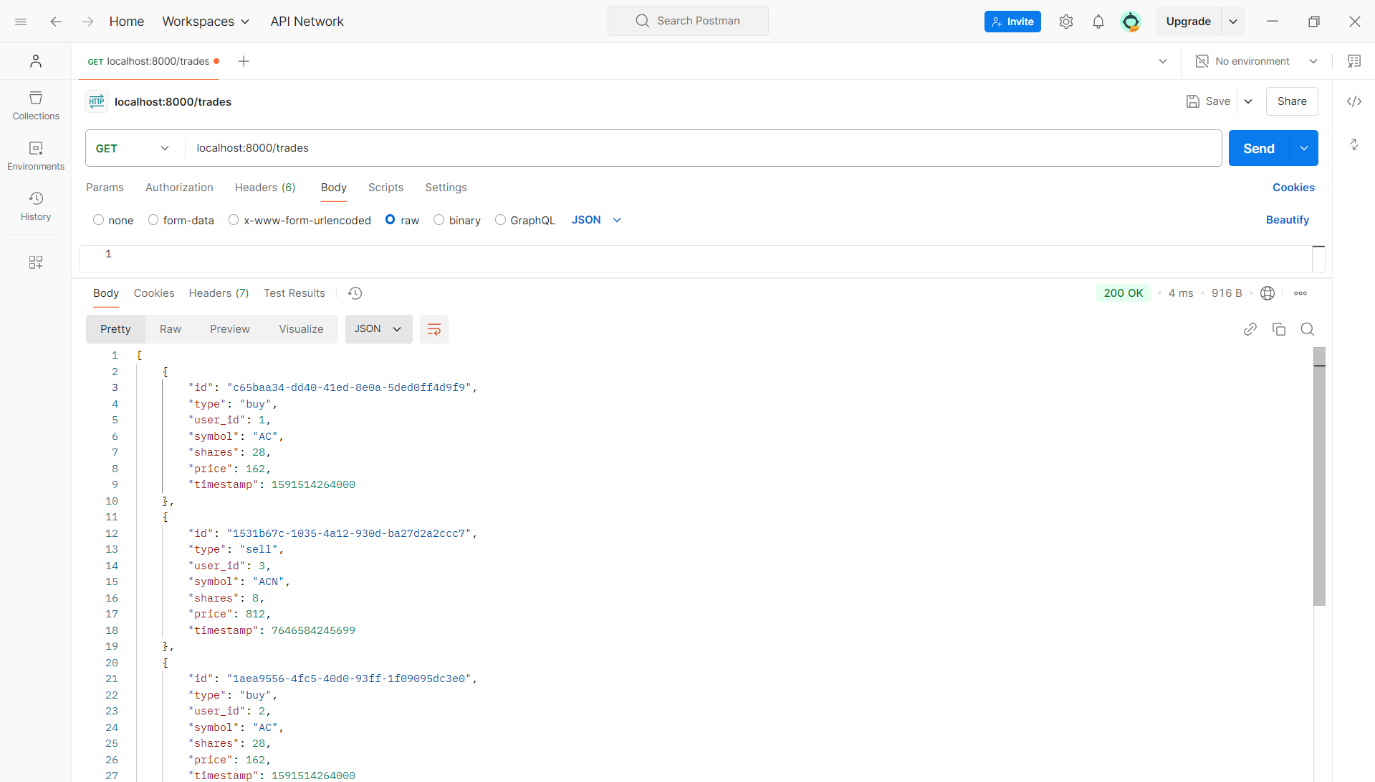
app.listen(PORT, () => {

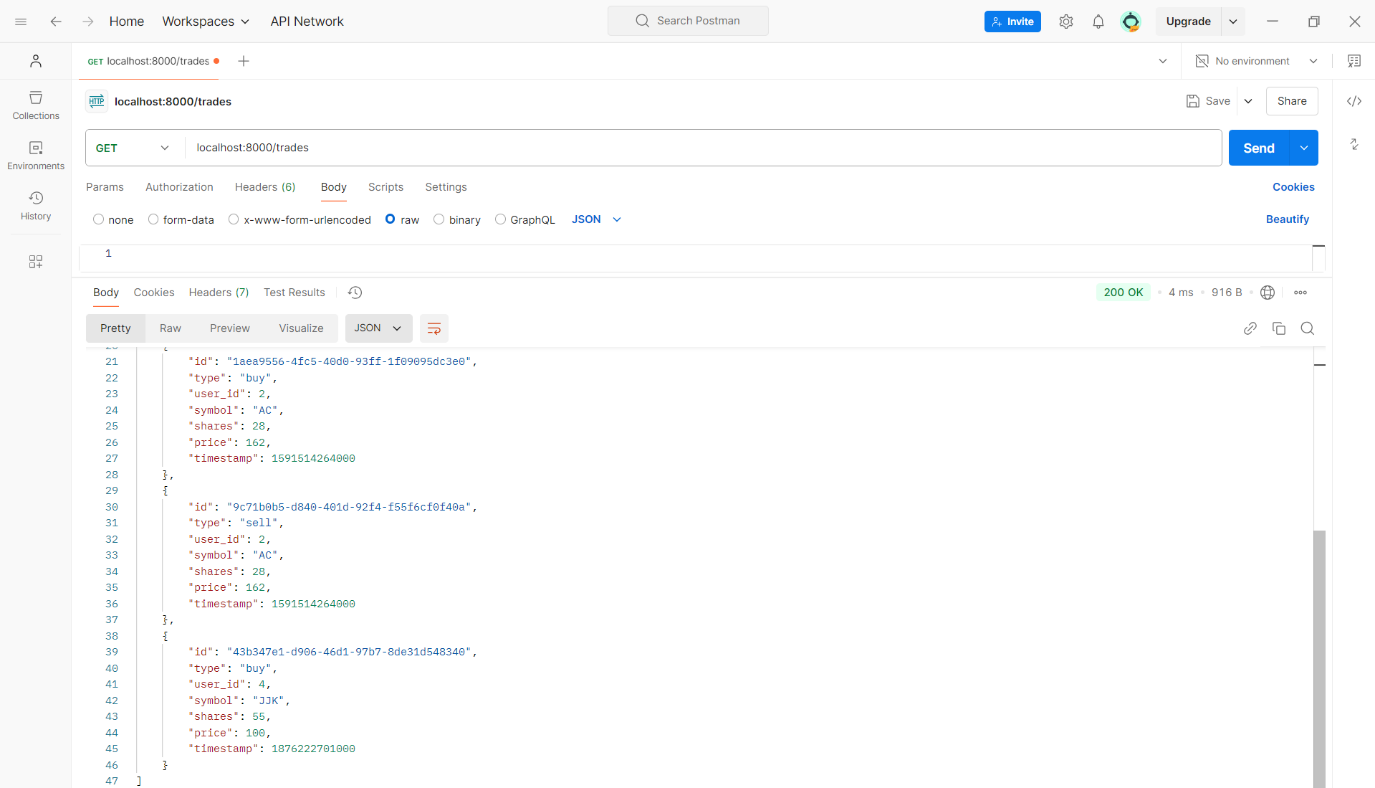
    console.log(`Server running on port ${PORT}`);

});

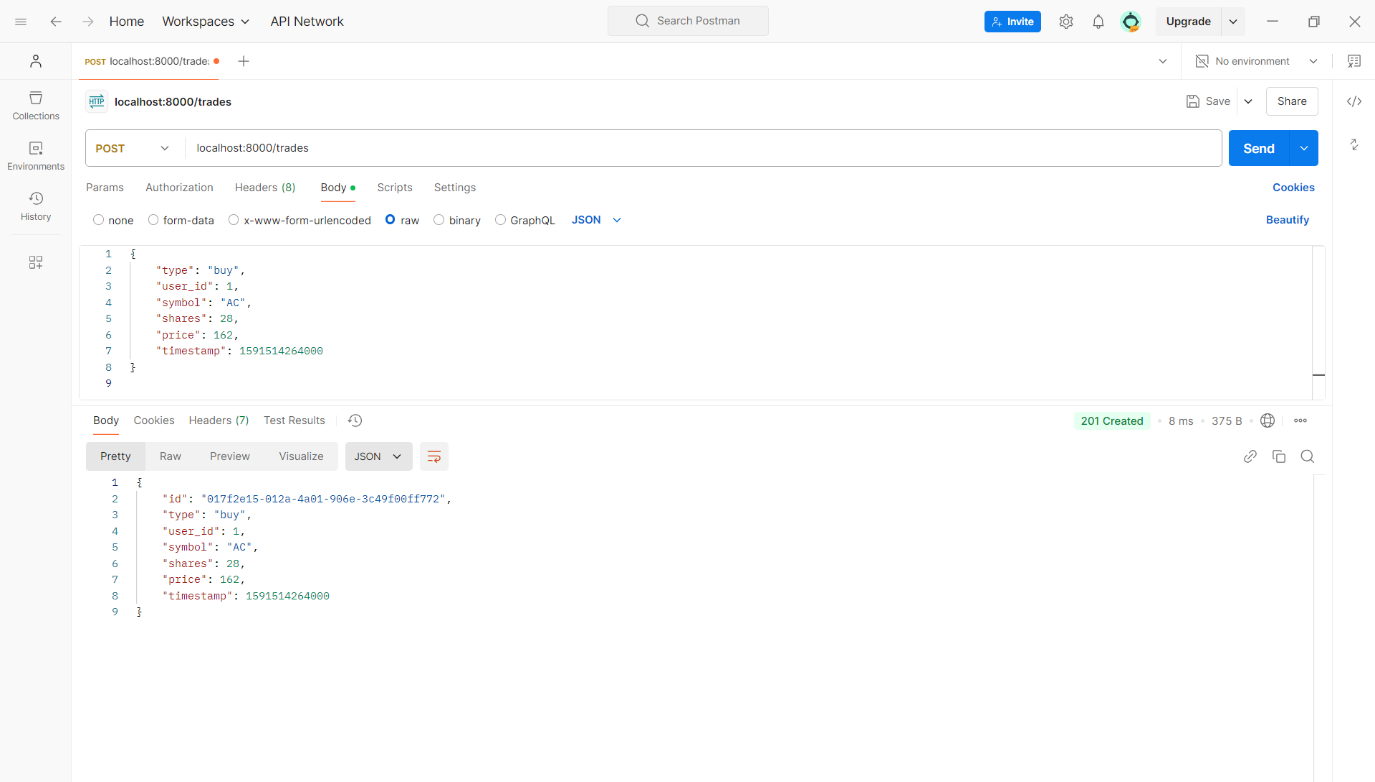
OUTPUT:

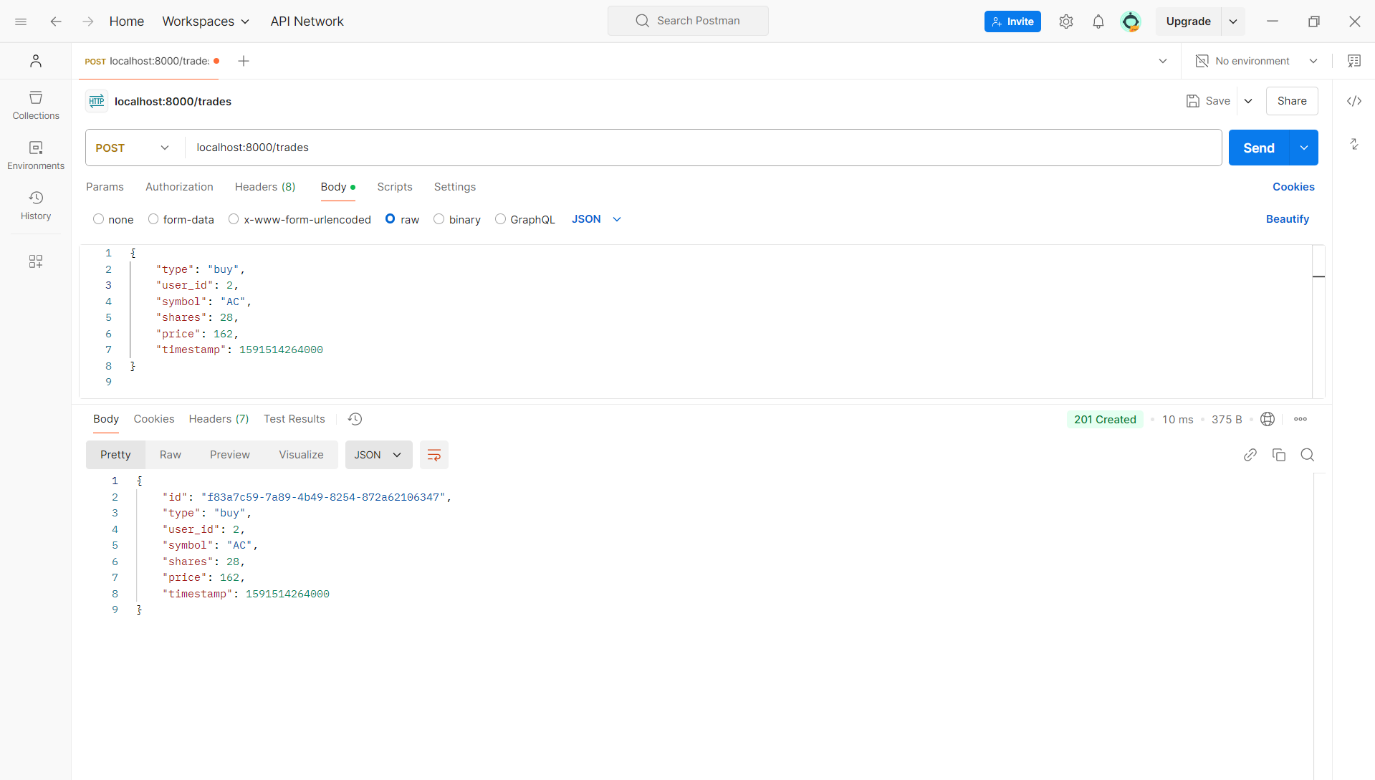
1. Each Trade in JSON Entry

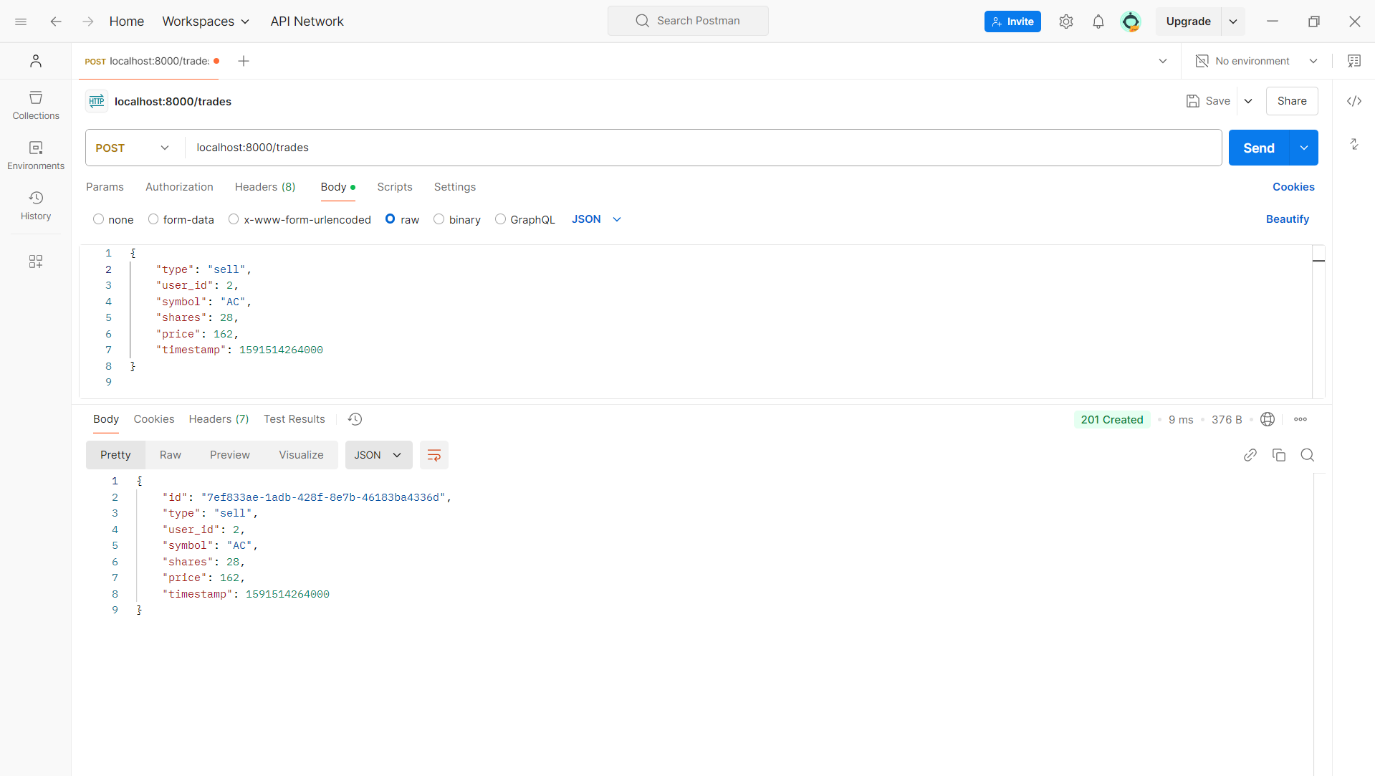


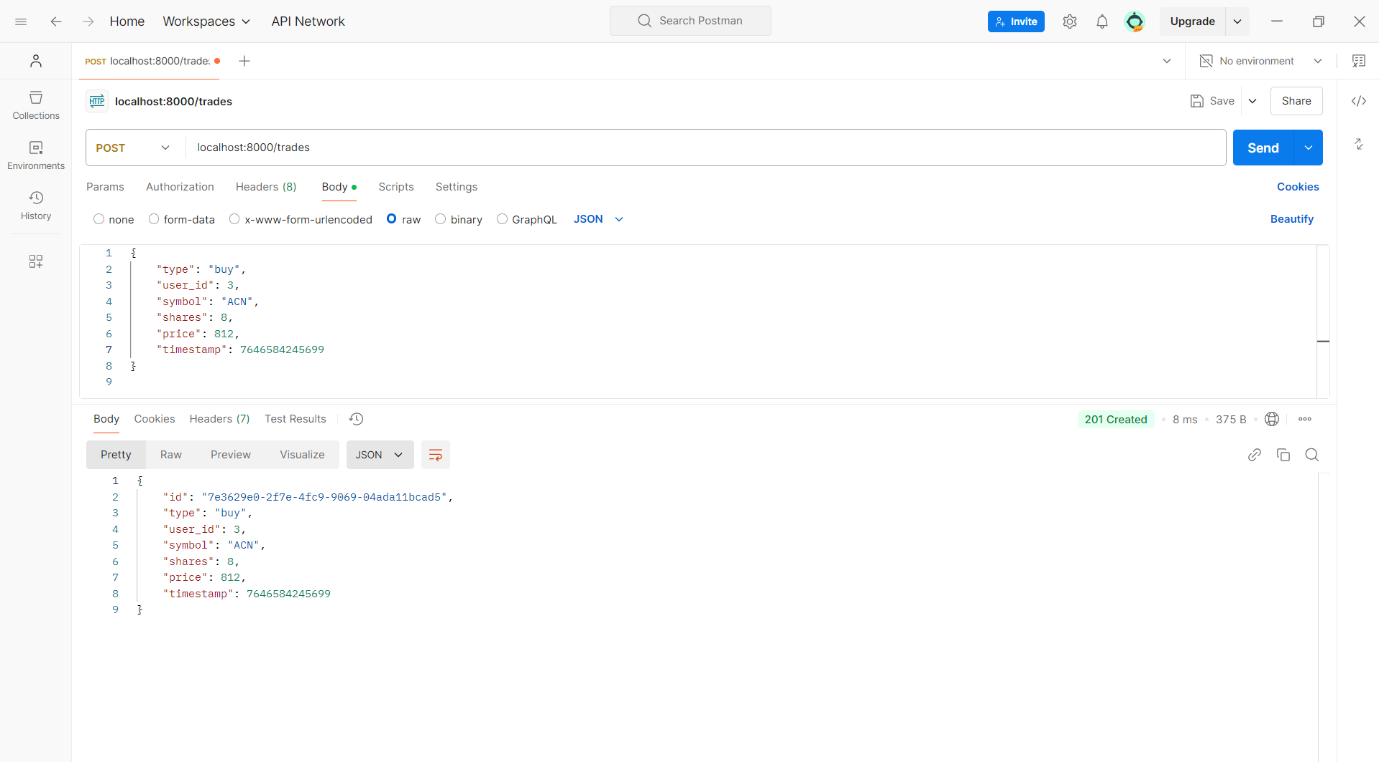


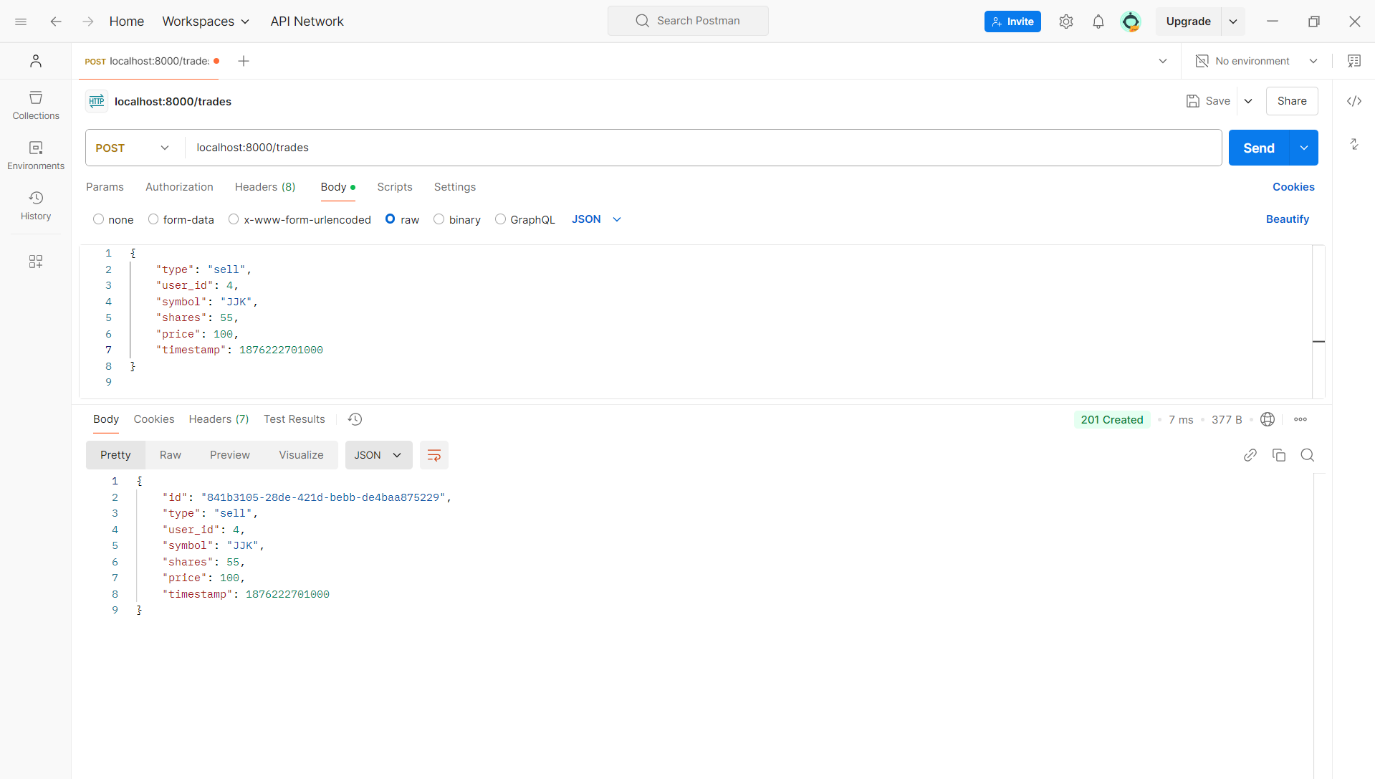
2. POST request to trades



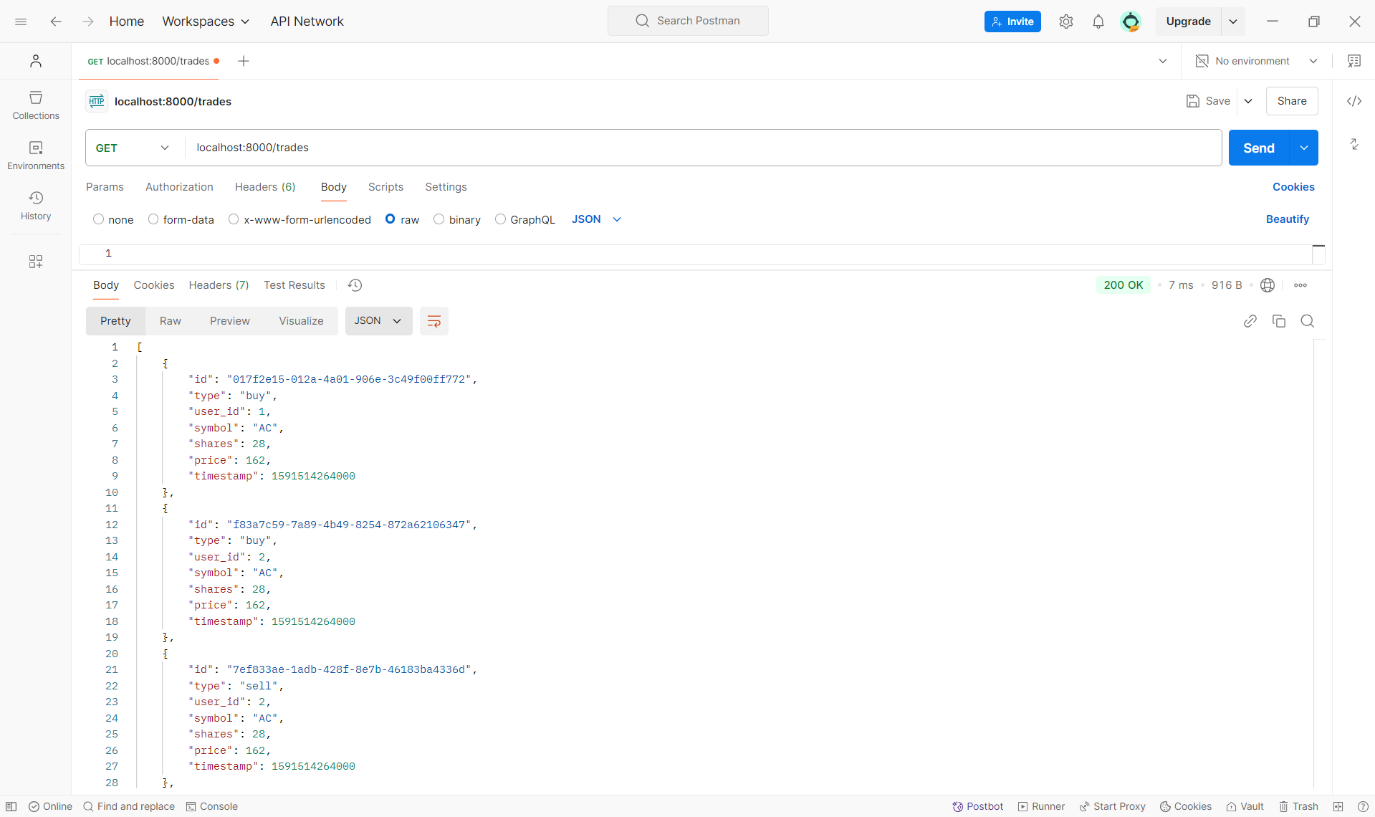


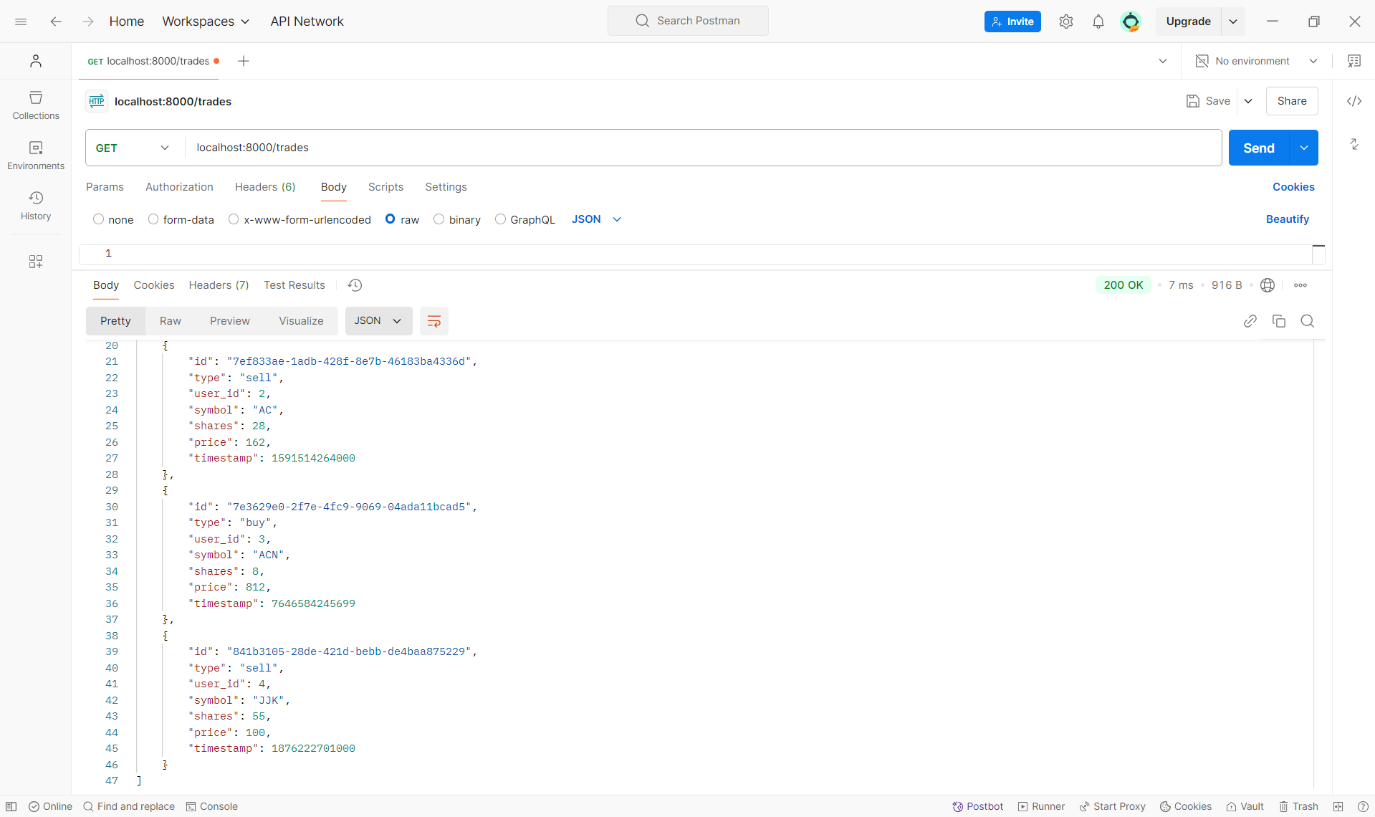




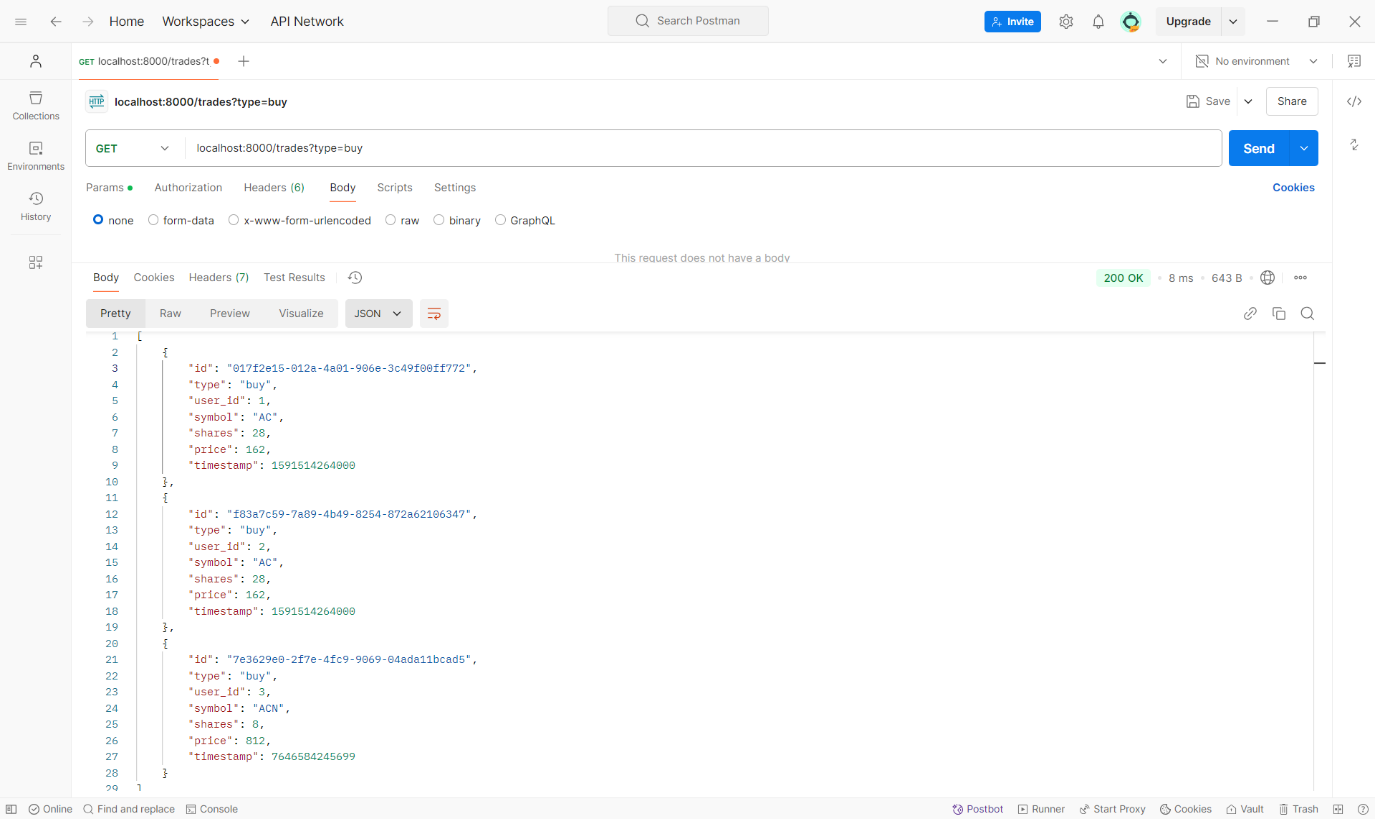


3. GET request to Trades

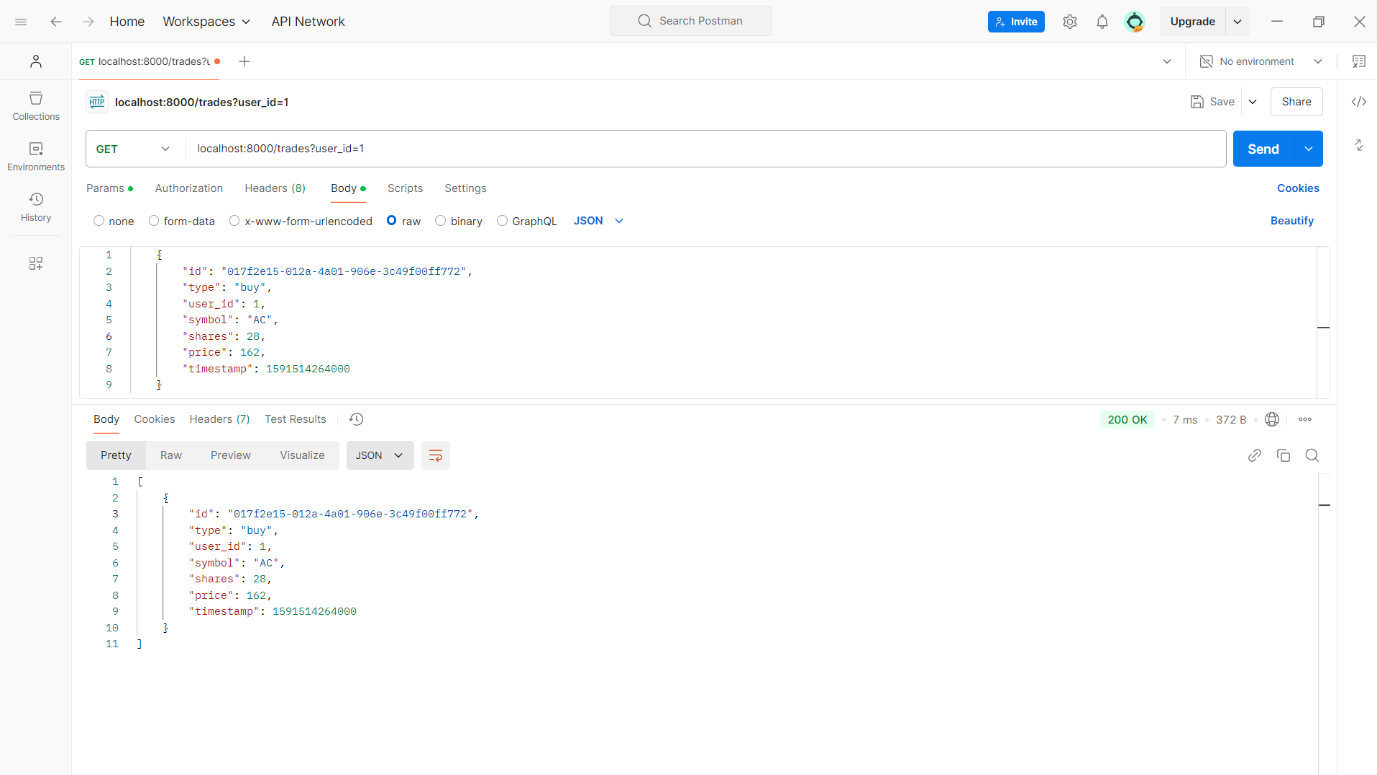


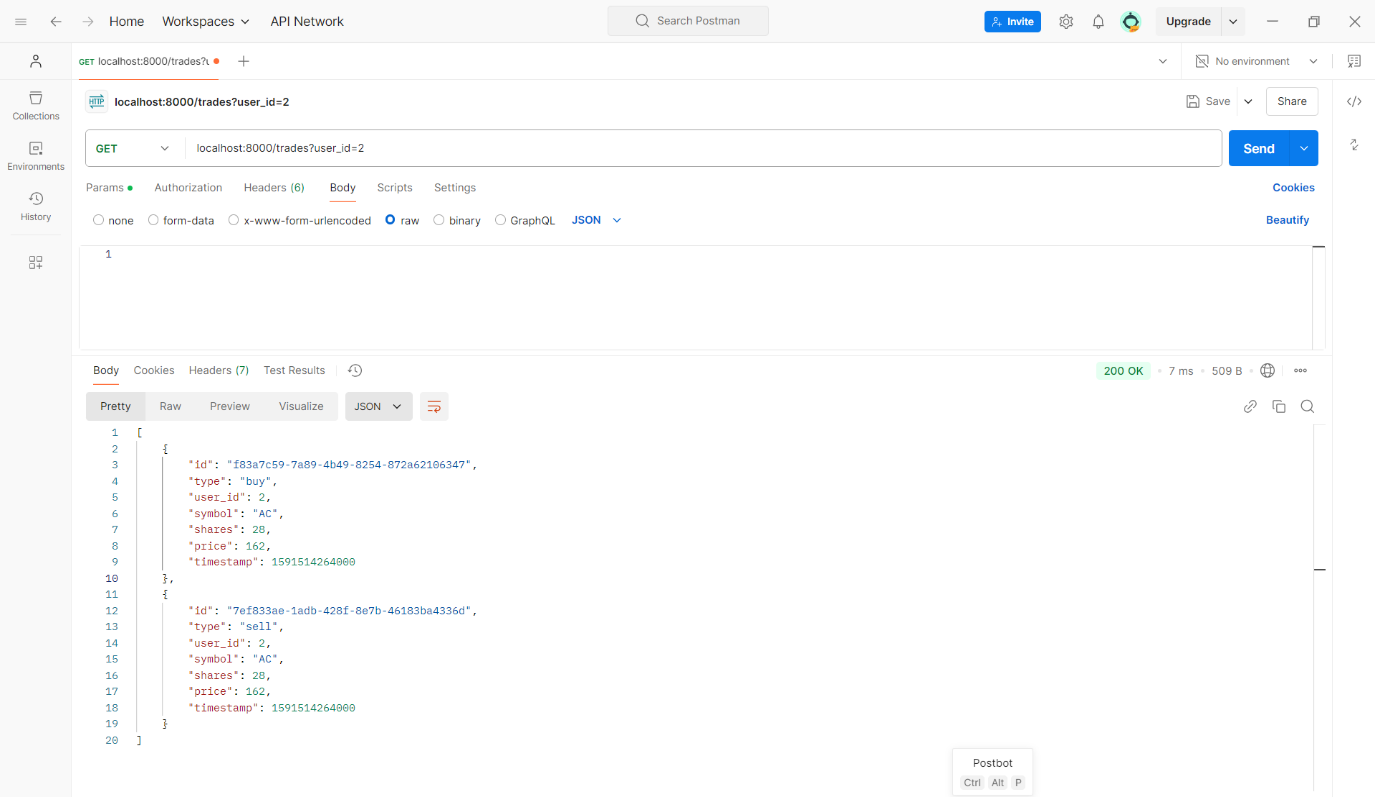


4. GET request to /trades?type=buy

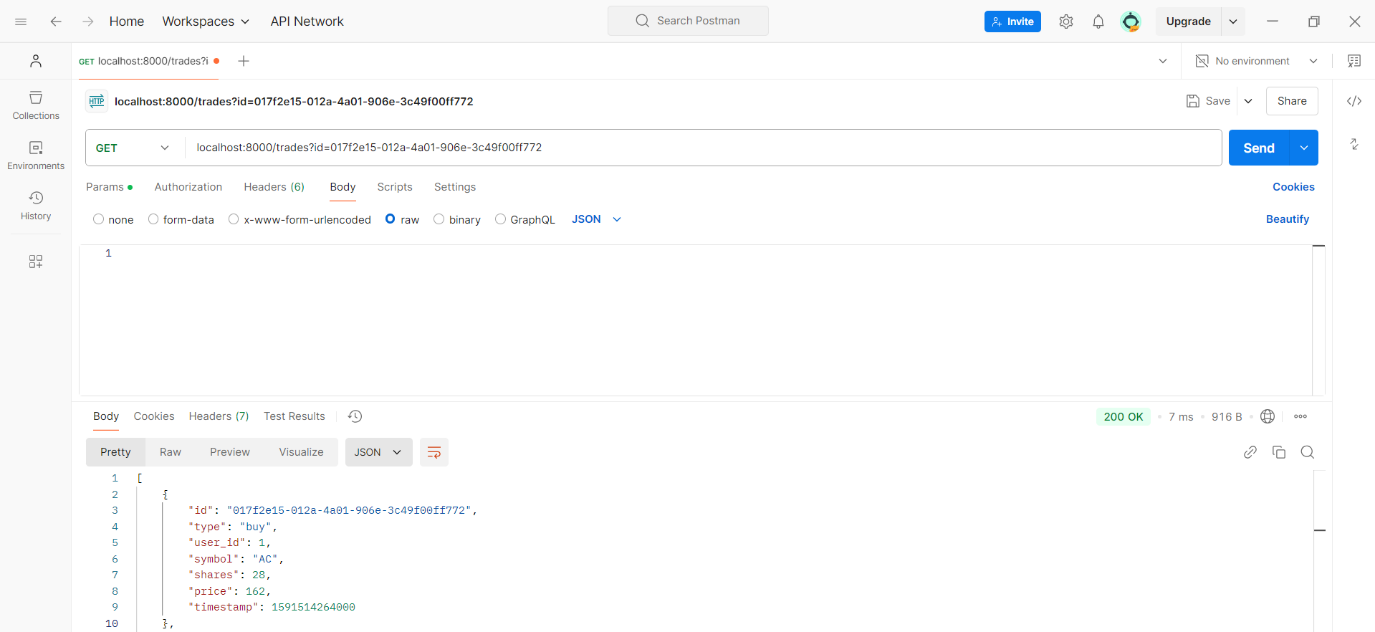


5. GET request to /trades?user\_id=2





6. GET request to /trades/:id



7. DELETE, PATCH request to /trades/:id

