DTS NodeJS Capstone Assignment

Name: Atharva Athanikar 72287317E

Batch: DTS Batch-3

Structure:

```
1) Stocks-trade-api
      i) Config
            (a)   db.cjs
      ii)   controllers
            (a)   tradeController.cjs
      iii)  middlewares
            (a)   authMiddleware.cjs
      iv)   models
            (a)   trade.cjs
      v) routes
            (a)   tradeRoutes.cjs
      vi)   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

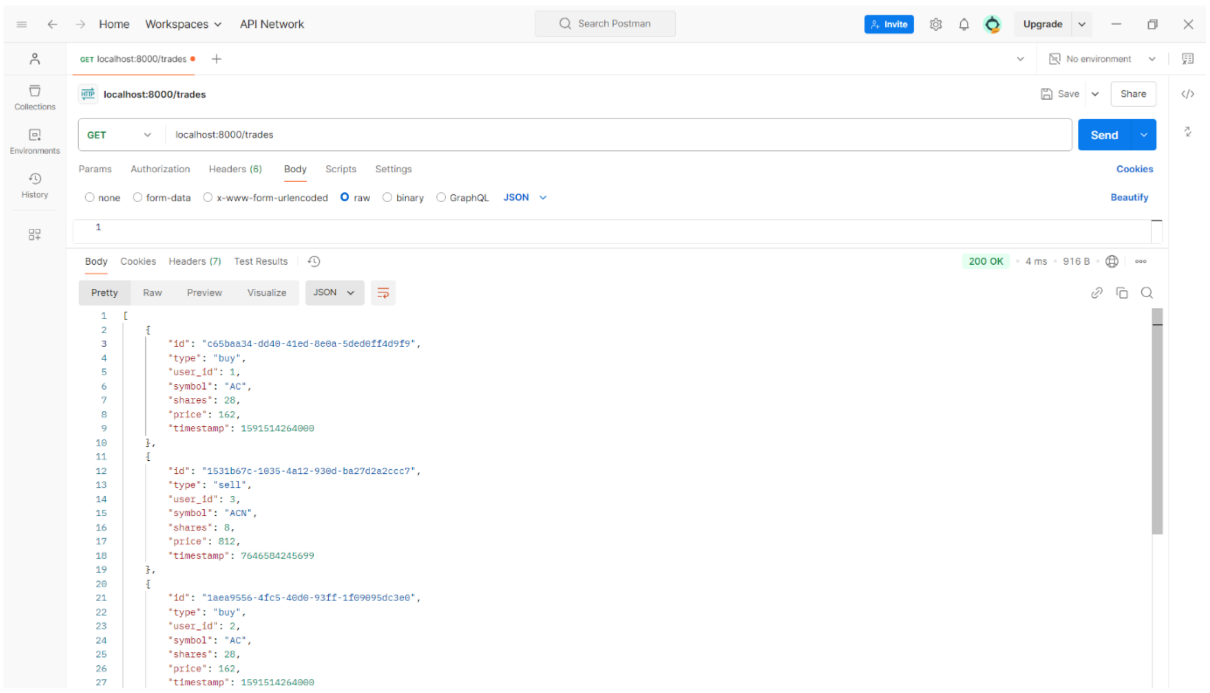
```
const { Router } = require("express");
const router = Router();
const { createTrade, getAllTrades, getTradeById } =
require("../controllers/tradeController.cjs");
const authMiddleware = require("../middlewares/authMiddleware.cjs");

// 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

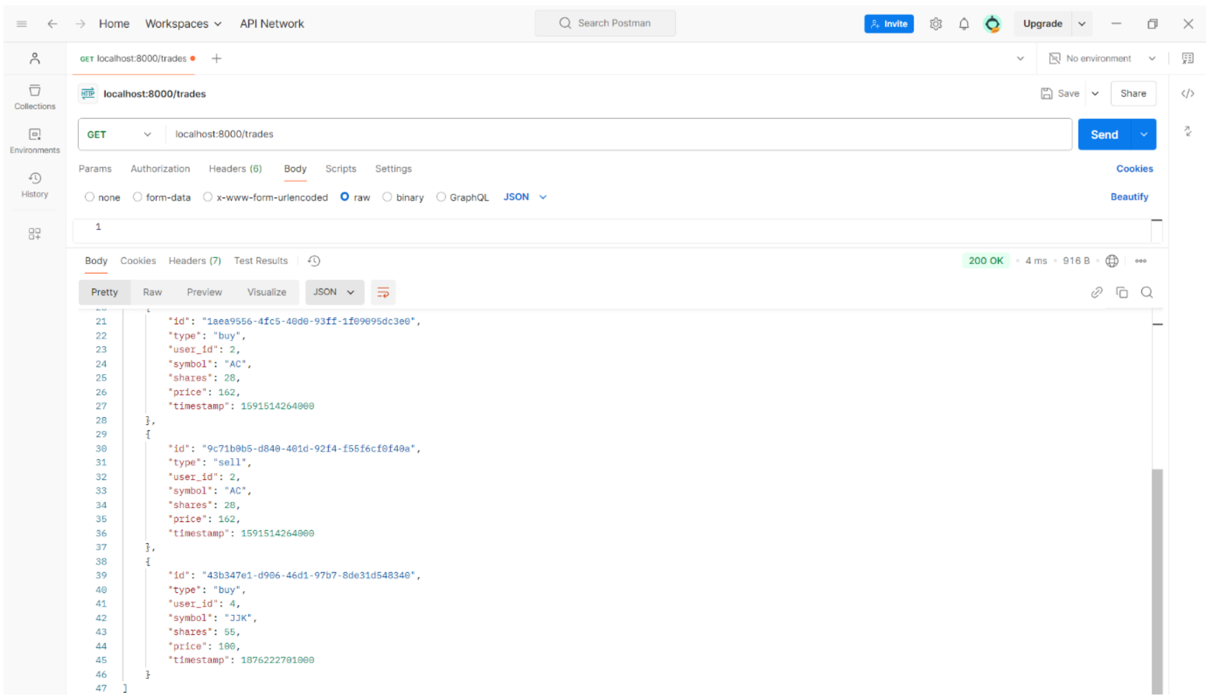
```
const express = require("express");
const { MongoClient } = require("mongodb");
const { v4: uuidv4 } = require("uuid");

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

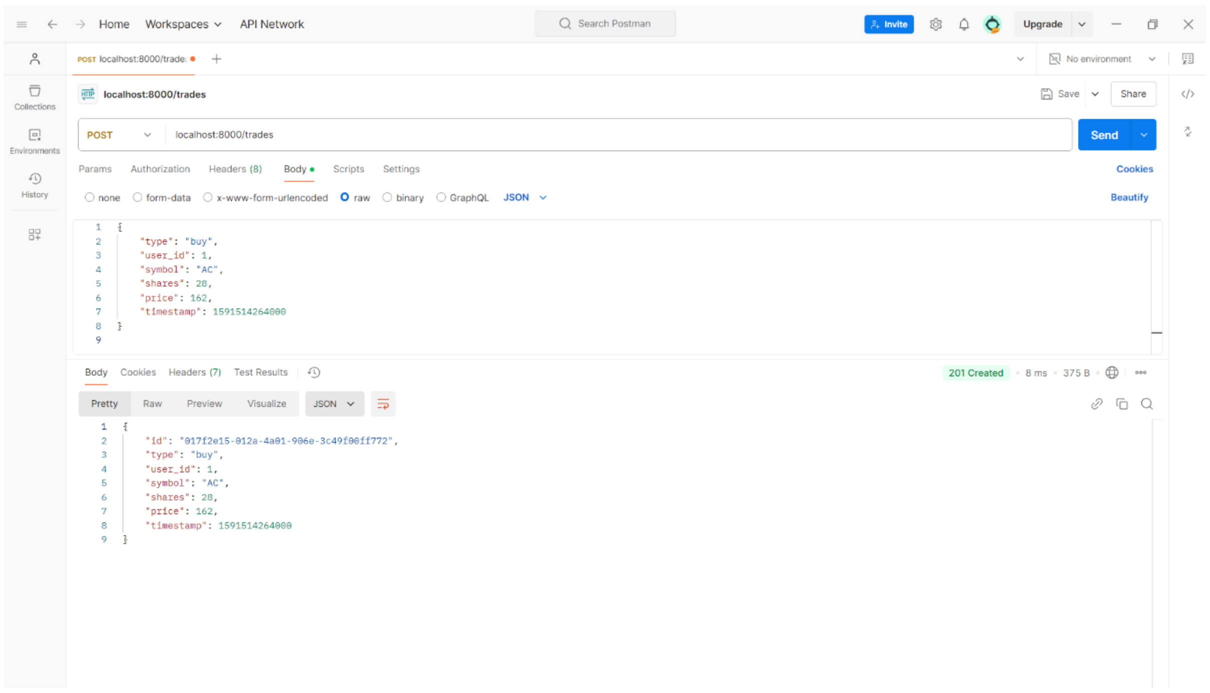

GET localhost:8000/trades

```
[
    {
        "id": "c65baa34-dd40-41ed-8e0a-5ded0ff4d9f9",
        "type": "buy",
        "user_id": 1,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    },
    {
        "id": "1531b67c-1035-4a12-930d-ba27d2a2ccc7",
        "type": "sell",
        "user_id": 3,
        "symbol": "ACN",
        "shares": 8,
        "price": 812,
        "timestamp": 7646584245699
    },
    {
        "id": "1aea9556-4fc5-40d0-93ff-1f09095dc3e0",
        "type": "buy",
        "user_id": 2,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
```


GET localhost:8000/trades

```
        "id": "1aea9556-4fc5-40d0-93ff-1f09095dc3e0",
        "type": "buy",
        "user_id": 2,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    },
    {
        "id": "9c71b0b5-d840-401d-92f4-f55f6cf0f40a",
        "type": "sell",
        "user_id": 2,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    },
    {
        "id": "43b347e1-d906-46d1-97b7-8de31d548340",
        "type": "buy",
        "user_id": 4,
        "symbol": "JJK",
        "shares": 55,
        "price": 100,
        "timestamp": 1876222701000
    }
]
```

2. POST request to trades

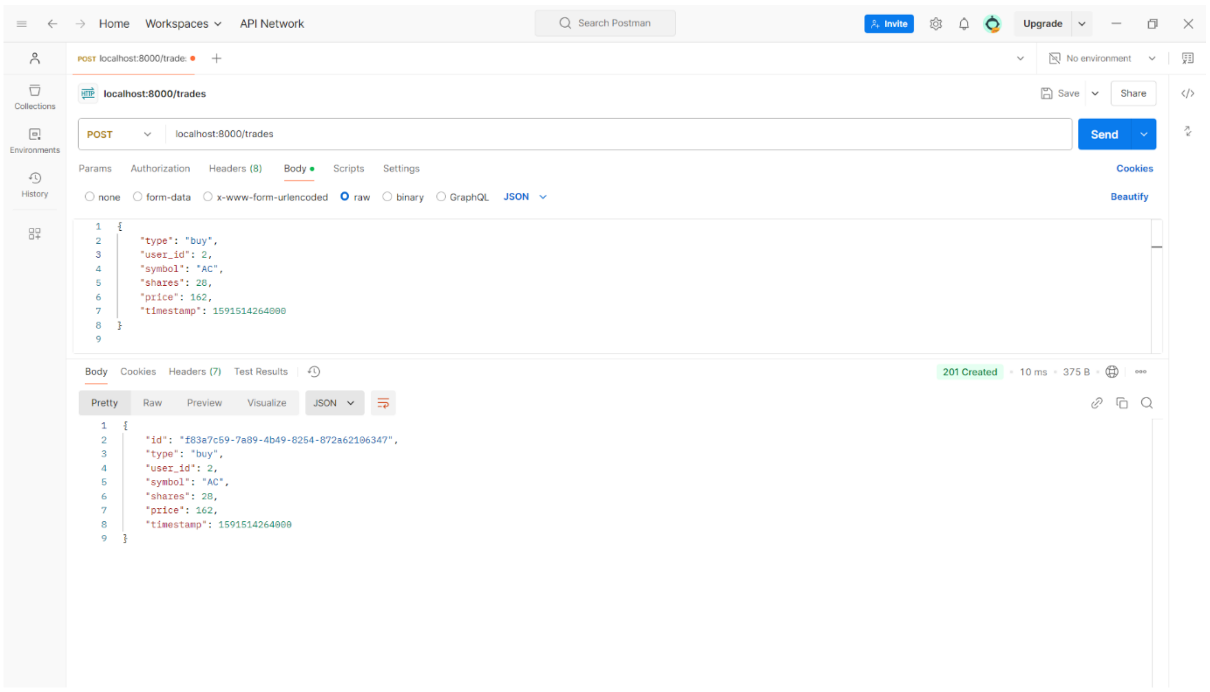


POST localhost:8000/trades

```
{
    "type": "buy",
    "user_id": 1,
    "symbol": "AC",
    "shares": 28,
    "price": 162,
    "timestamp": 1591514264000
}
```

201 Created · 8 ms · 375 B

```
{
    "id": "017f2e15-012a-4a01-906e-3c49f00ff772",
    "type": "buy",
    "user_id": 1,
    "symbol": "AC",
    "shares": 28,
    "price": 162,
    "timestamp": 1591514264000
}
```

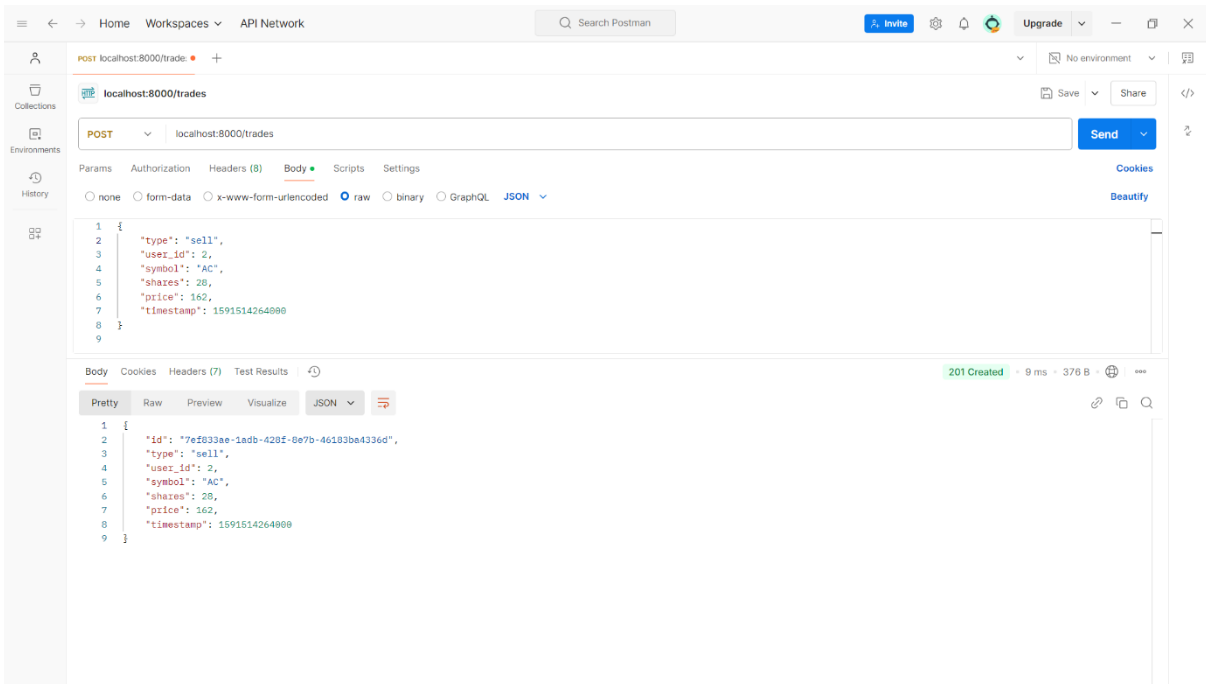


POST localhost:8000/trades

```
{
    "type": "buy",
    "user_id": 2,
    "symbol": "AC",
    "shares": 28,
    "price": 162,
    "timestamp": 1591514264000
}
```

201 Created · 10 ms · 375 B

```
{
    "id": "f83a7c59-7a89-4b49-8254-872a62106347",
    "type": "buy",
    "user_id": 2,
    "symbol": "AC",
    "shares": 28,
    "price": 162,
    "timestamp": 1591514264000
}
```



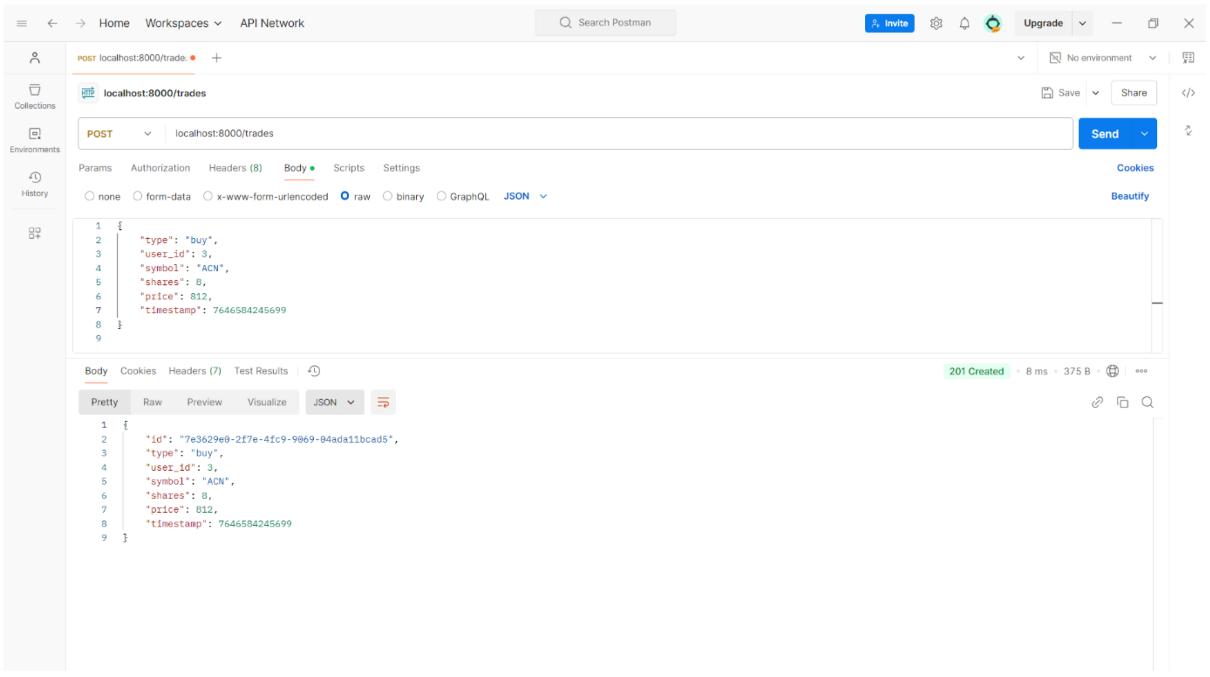
POST localhost:8000/trades

Body: none / form-data / x-www-form-urlencoded / [✓ raw] / binary / GraphQL — JSON

```
{
    "type": "sell",
    "user_id": 2,
    "symbol": "AC",
    "shares": 28,
    "price": 162,
    "timestamp": 1591514264000
}
```

201 Created · 9 ms · 376 B

```
{
    "id": "7ef833ae-1adb-428f-8e7b-46183ba4336d",
    "type": "sell",
    "user_id": 2,
    "symbol": "AC",
    "shares": 28,
    "price": 162,
    "timestamp": 1591514264000
}
```



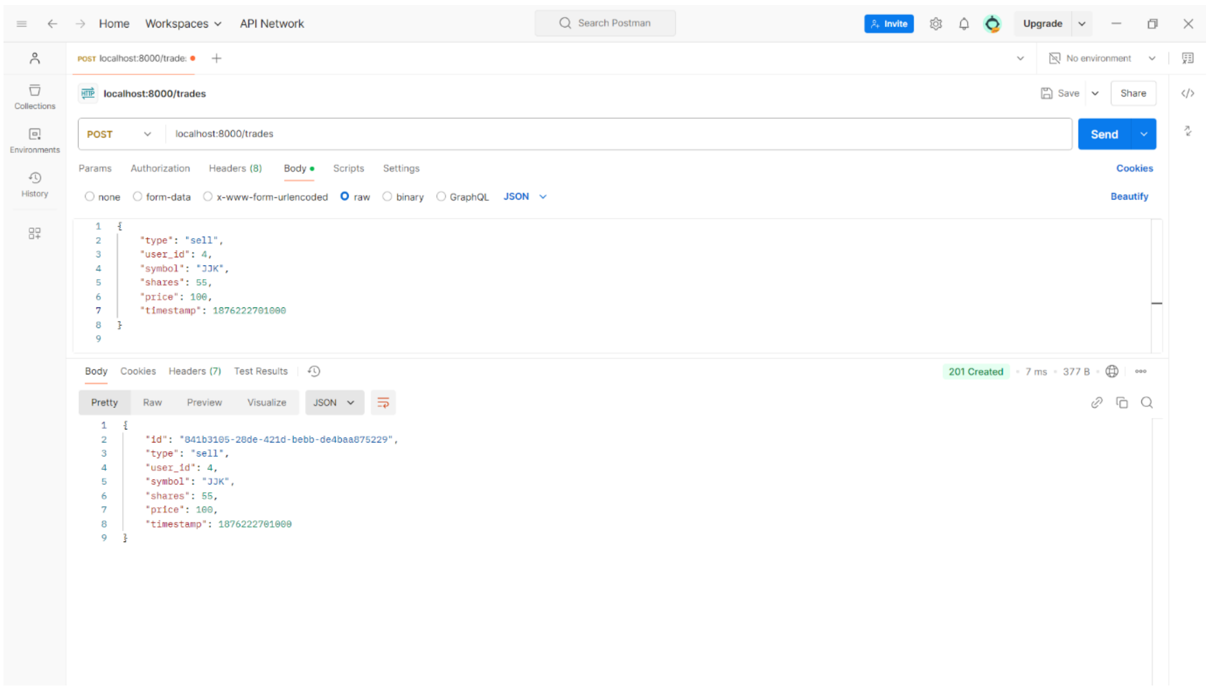
POST localhost:8000/trades

Body: none / form-data / x-www-form-urlencoded / [✓ raw] / binary / GraphQL — JSON

```
{
    "type": "buy",
    "user_id": 3,
    "symbol": "ACN",
    "shares": 8,
    "price": 812,
    "timestamp": 7646584245699
}
```

201 Created · 8 ms · 375 B

```
{
    "id": "7e3629e0-2f7e-4fc9-9069-04ada11bcad5",
    "type": "buy",
    "user_id": 3,
    "symbol": "ACN",
    "shares": 8,
    "price": 812,
    "timestamp": 7646584245699
}
```

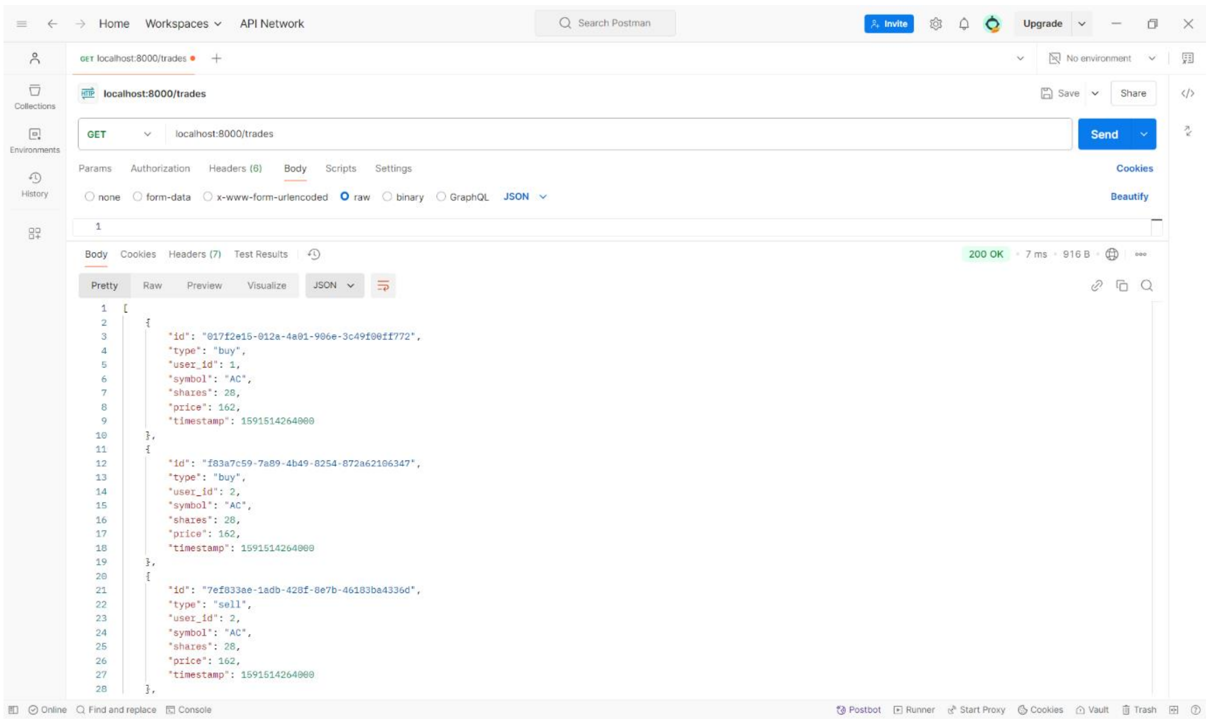

POST localhost:8000/trades

```
{
    "type": "sell",
    "user_id": 4,
    "symbol": "JJK",
    "shares": 55,
    "price": 100,
    "timestamp": 1876222701000
}
```

201 Created · 7 ms · 377 B

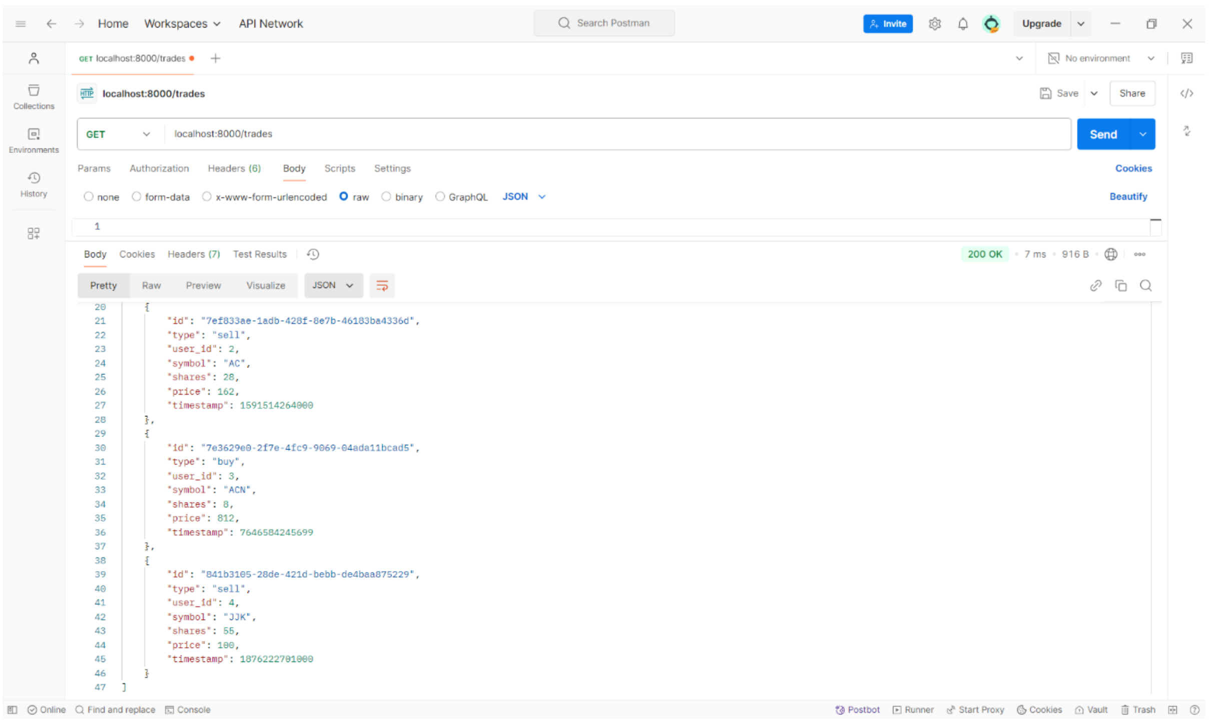
```
{
    "id": "841b3105-28de-421d-bebb-de4baa875229",
    "type": "sell",
    "user_id": 4,
    "symbol": "JJK",
    "shares": 55,
    "price": 100,
    "timestamp": 1876222701000
}
```

3. GET request to Trades


GET localhost:8000/trades

200 OK · 7 ms · 916 B

```
[
    {
        "id": "017f2e15-012a-4a01-906e-3c49f00ff772",
        "type": "buy",
        "user_id": 1,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    },
    {
        "id": "f83a7c59-7a89-4b49-8254-872a62106347",
        "type": "buy",
        "user_id": 2,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    },
    {
        "id": "7ef833ae-1adb-428f-8e7b-46183ba4336d",
        "type": "sell",
        "user_id": 2,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    },
```

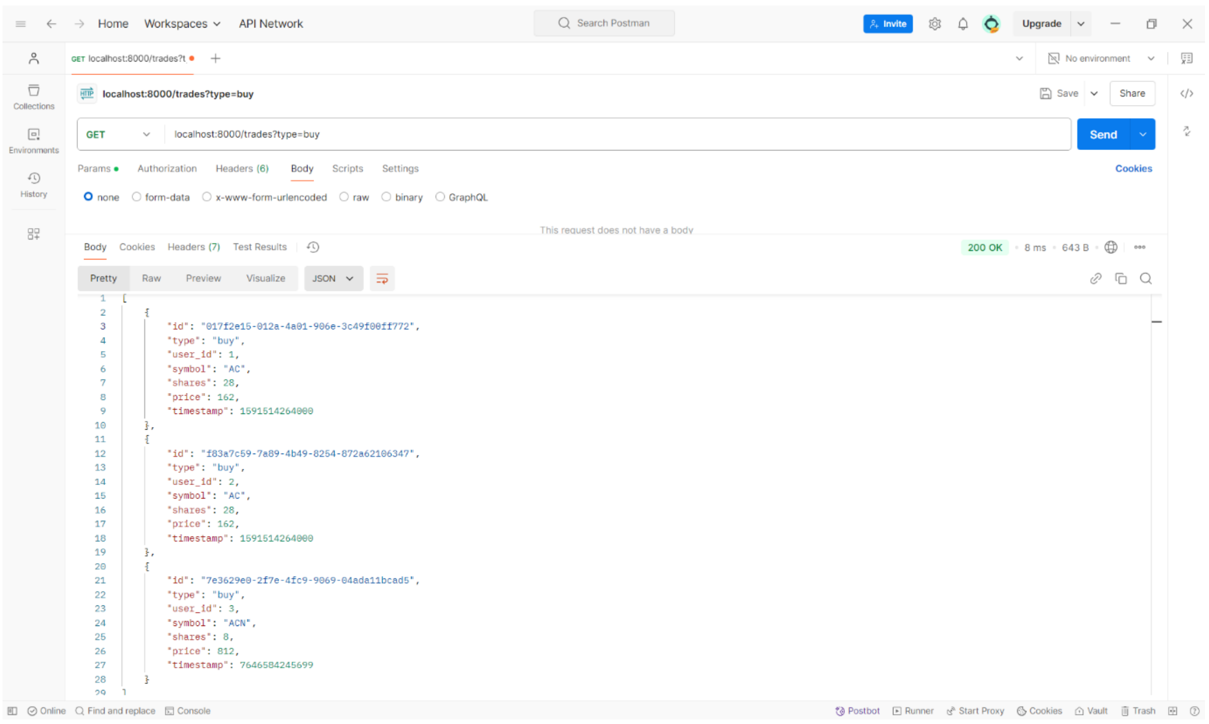


GET localhost:8000/trades

200 OK · 7 ms · 916 B

```
    {
        "id": "7ef833ae-1adb-428f-8e7b-46183ba4336d",
        "type": "sell",
        "user_id": 2,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    },
    {
        "id": "7e3629e0-2f7e-4fc9-9069-04ada11bcad5",
        "type": "buy",
        "user_id": 3,
        "symbol": "ACN",
        "shares": 8,
        "price": 812,
        "timestamp": 7646584245699
    },
    {
        "id": "841b3105-28de-421d-bebb-de4baa875229",
        "type": "sell",
        "user_id": 4,
        "symbol": "JJK",
        "shares": 55,
        "price": 100,
        "timestamp": 1876222701000
    }
]
```

4. GET request to /trades?type=buy



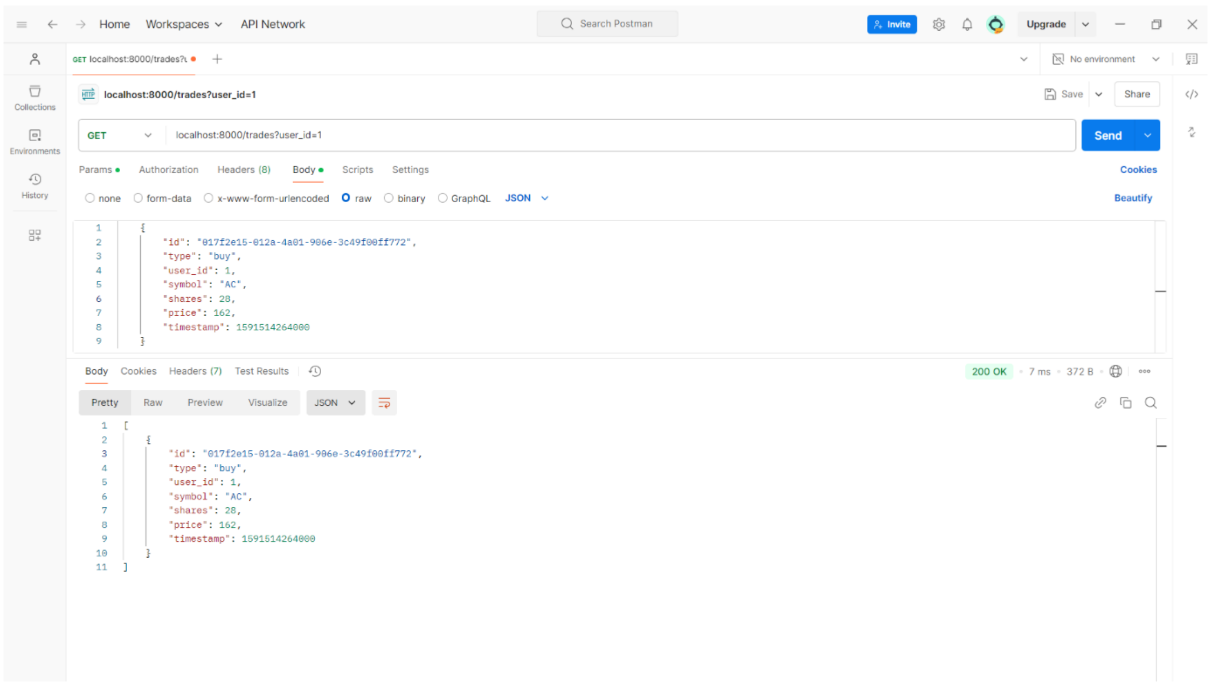
GET localhost:8000/trades?type=buy

This request does not have a body

200 OK · 8 ms · 643 B

```
[
    {
        "id": "017f2e15-012a-4a01-906e-3c49f00ff772",
        "type": "buy",
        "user_id": 1,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    },
    {
        "id": "f83a7c59-7a89-4b49-8254-872a62106347",
        "type": "buy",
        "user_id": 2,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    },
    {
        "id": "7e3629e0-2f7e-4fc9-9069-04ada11bcad5",
        "type": "buy",
        "user_id": 3,
        "symbol": "ACN",
        "shares": 8,
        "price": 812,
        "timestamp": 7646584245699
    }
]
```

5. GET request to /trades?user_id=2



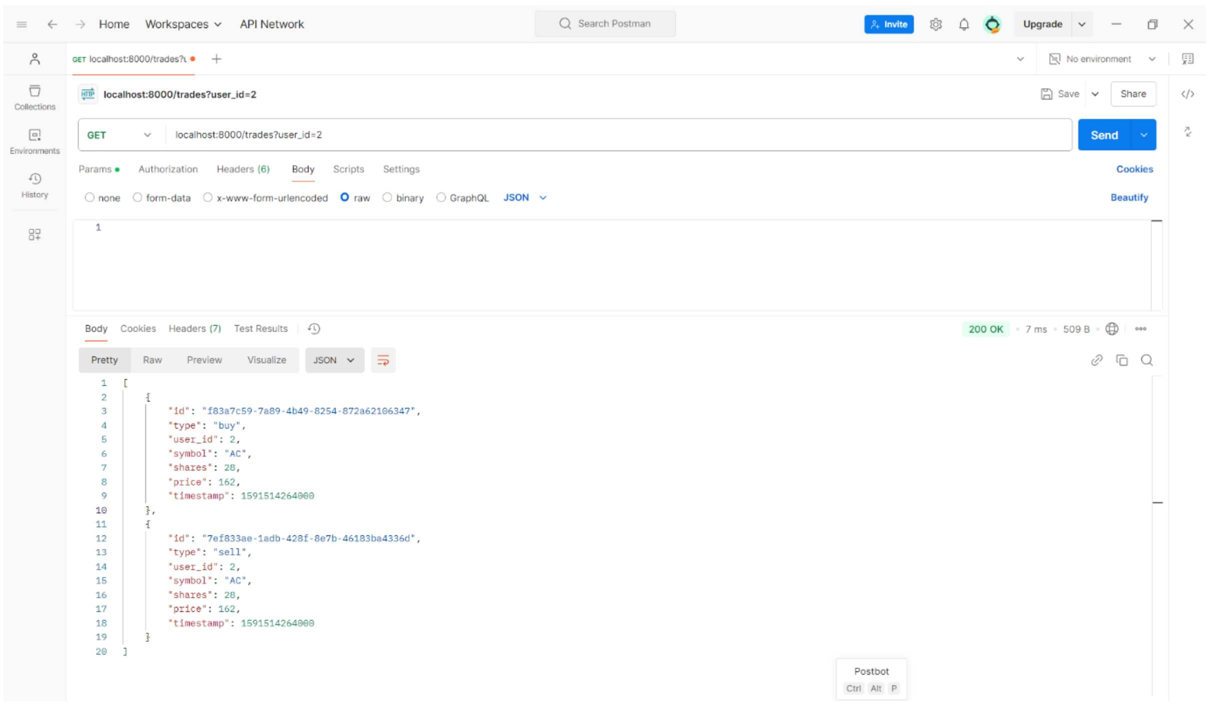
GET localhost:8000/trades?user_id=1

Body: raw JSON

```
{
    "id": "017f2e15-012a-4a01-906e-3c49f00ff772",
    "type": "buy",
    "user_id": 1,
    "symbol": "AC",
    "shares": 28,
    "price": 162,
    "timestamp": 1591514264000
}
```

200 OK · 7 ms · 372 B

```
[
    {
        "id": "017f2e15-012a-4a01-906e-3c49f00ff772",
        "type": "buy",
        "user_id": 1,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    }
]
```



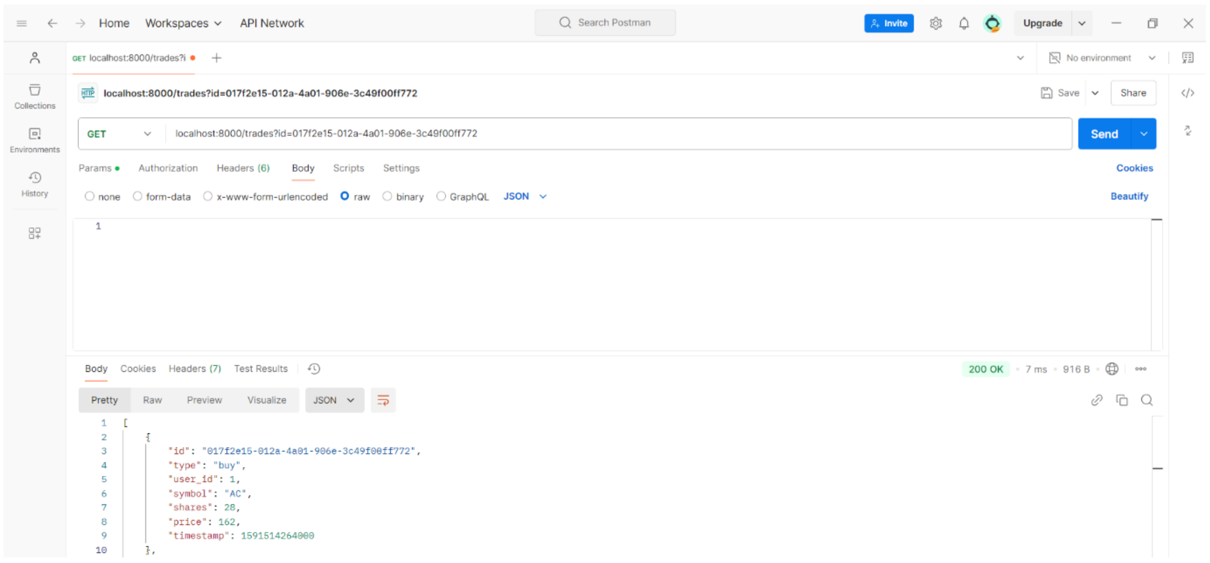
GET localhost:8000/trades?user_id=2

Body: raw JSON

200 OK · 7 ms · 509 B

```
[
    {
        "id": "f83a7c59-7a89-4b49-8254-872a62106347",
        "type": "buy",
        "user_id": 2,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    },
    {
        "id": "7ef833ae-1adb-428f-8e7b-46183ba4336d",
        "type": "sell",
        "user_id": 2,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    }
]
```

6. GET request to /trades/:id



localhost:8000/trades?id=017f2e15-012a-4a01-906e-3c49f00ff772

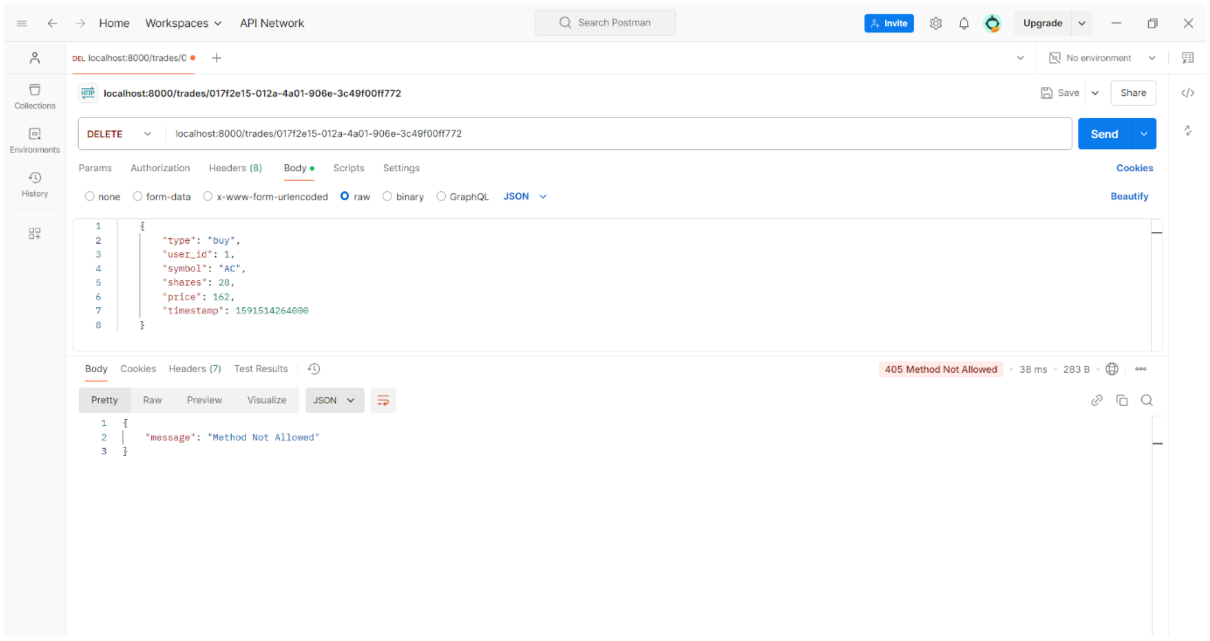
GET localhost:8000/trades?id=017f2e15-012a-4a01-906e-3c49f00ff772

Body: raw JSON

200 OK · 7 ms · 916 B

```
[
    {
        "id": "017f2e15-012a-4a01-906e-3c49f00ff772",
        "type": "buy",
        "user_id": 1,
        "symbol": "AC",
        "shares": 28,
        "price": 162,
        "timestamp": 1591514264000
    },
```

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



localhost:8000/trades/017f2e15-012a-4a01-906e-3c49f00ff772

DELETE localhost:8000/trades/017f2e15-012a-4a01-906e-3c49f00ff772

Body: raw JSON

```
{
    "type": "buy",
    "user_id": 1,
    "symbol": "AC",
    "shares": 28,
    "price": 162,
    "timestamp": 1591514264000
}
```

405 Method Not Allowed · 38 ms · 283 B

```
{
    "message": "Method Not Allowed"
}
```