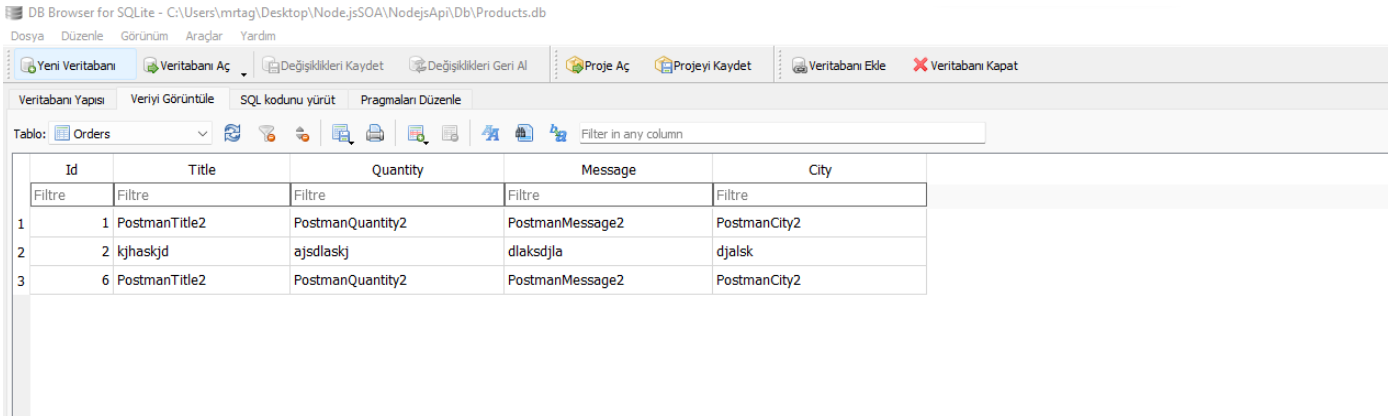
SOA PROJE ÖDEVİ

İSTERLER

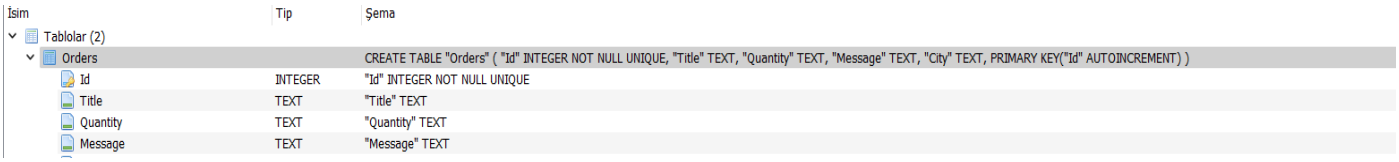
Database:

Veriler:

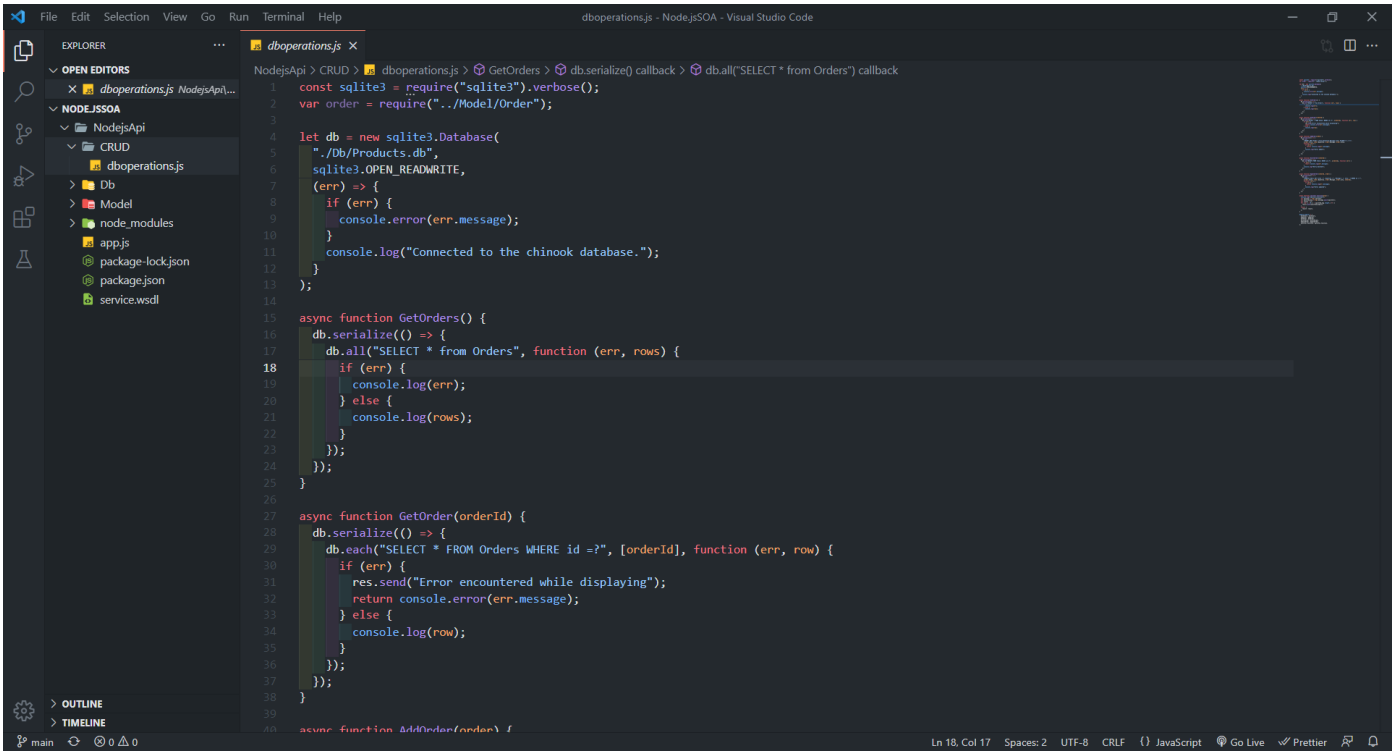


	Id	Title	Quantity	Message	City
	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	PostmanTitle2	PostmanQuantity2	PostmanMessage2	PostmanCity2
2	2	kjhaskjd	ajsdlaskj	dlaksdjla	djalsk
3	6	PostmanTitle2	PostmanQuantity2	PostmanMessage2	PostmanCity2

Veritabanı Yapısı:



Backend:



- **Dboperations.js :**

```
const sqlite3 = require("sqlite3").verbose();
var order = require("../Model/Order");


let db = new sqlite3.Database(
  "./Db/Products.db",
  sqlite3.OPEN_READWRITE,
  (err) => {
    if (err) {
      console.error(err.message);
    }
    console.log("Connected to the chinook database.");
  }
);

async function GetOrders() {
  db.serialize(() => {
    db.all("SELECT * from Orders", function (err, rows) {
      if (err) {
        console.log(err);
      } else {
        console.log(rows);
      }
    });
  });
}
async function GetOrder(orderId) {
  db.serialize(() => {
    db.each("SELECT * FROM Orders WHERE id =?", [orderId], function (err, row)
{
      if (err) {
```

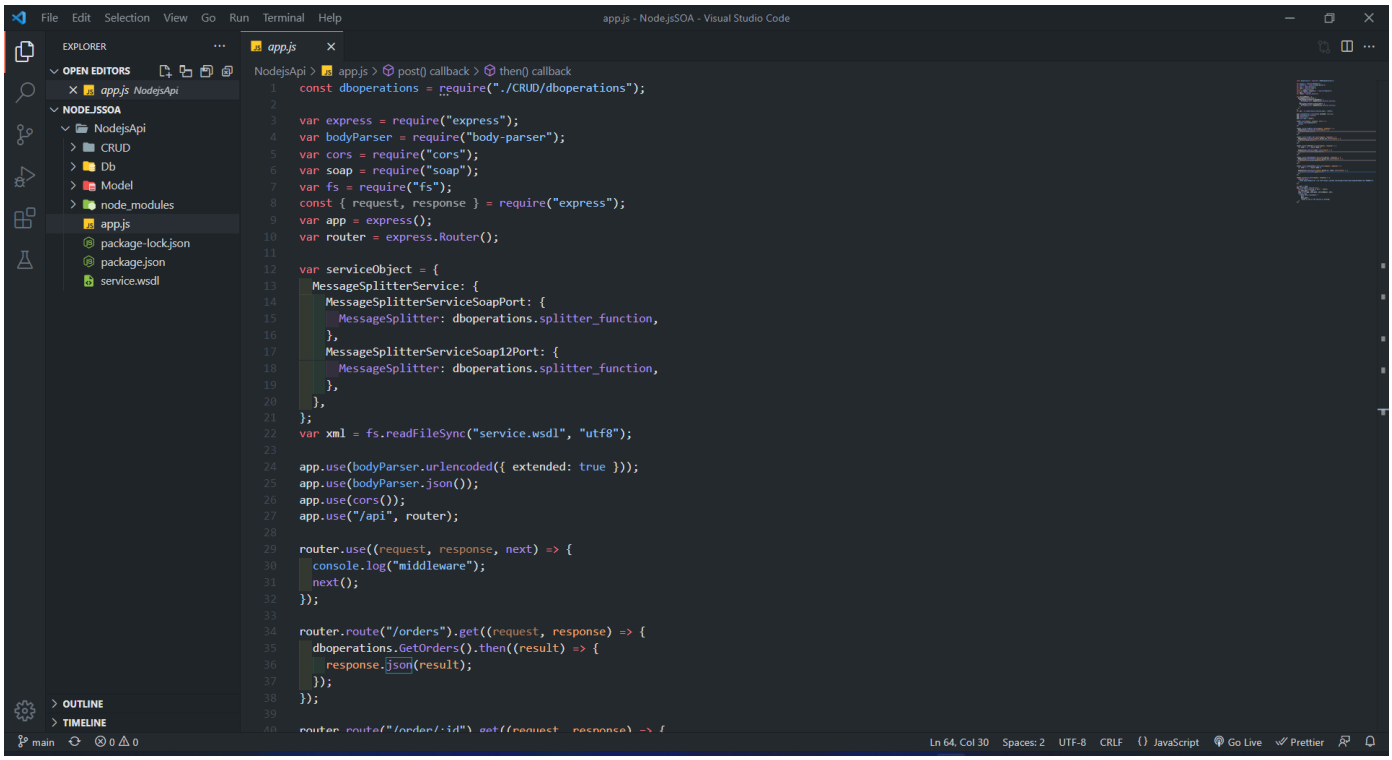
```
        res.send("Error encountered while displaying");
        return console.error(err.message);
      } else {
        console.log(row);
      }
    });
  });
}
async function AddOrder(order) {
  db.serialize(() => {
    db.run(
      "INSERT INTO Orders (Title,Quantity,Message,City) VALUES(?,?,?,?)",
      [order.Title, order.Quantity, order.Message, order.City],
      function (err) {
        if (err) {
          return console.log(err.message);
        }
        console.log("Entry added");
      }
    );
  });
}
async function DeleteOrder(orderId) {
  db.serialize(() => {
    db.run("DELETE FROM Orders WHERE id =?", [orderId], function (err) {
      if (err) {
        return console.log(err.message);
      }
      console.log("Entry deleted");
    });
  });
```

```
}
async function UpdateOrder(orderId, order) {
  db.serialize(() => {
    db.run(
      "UPDATE Orders SET Title = ?, Quantity = ?, Message = ?, City = ? WHERE
id = ?",
      [order.Title, order.Quantity, order.Message, order.City, orderId],
      function (err) {
        if (err) {
          return console.log(err.message);
        }
        console.log("Entry updated");
      }
    );
  });
}
async function splitter_function(args) {
  console.log("splitter_function");
  var splitter = args.splitter;
  var splitted_msg = args.message.split(splitter);
  var result = [];
  for (var i = 0; i < splitted_msg.length; i++) {
    result.push(splitted_msg[i]);
  }
  return {
    result: result,
  };
}


module.exports = {
  GetOrders: GetOrders,
```

```
  AddOrder: AddOrder,
  GetOrder: GetOrder,
  DeleteOrder: DeleteOrder,
  UpdateOrder: UpdateOrder,
  splitter_function: splitter_function,
};
```

API VE SOAP/WSDL:



- ***App.js:***

```
const dboperations = require("./CRUD/dboperations");
var express = require("express");
var bodyParser = require("body-parser");
var cors = require("cors");
var soap = require("soap");
var fs = require("fs");
const { request, response } = require("express");
var app = express();
```

```
var router = express.Router();

var serviceObject = {
  MessageSplitterService: {
    MessageSplitterServiceSoapPort: {
      MessageSplitter: dboperations.splitter_function,
    },
    MessageSplitterServiceSoap12Port: {
      MessageSplitter: dboperations.splitter_function,
    },
  },
};

var xml = fs.readFileSync("service.wsdl", "utf8");

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.use(cors());
app.use("/api", router);

router.use((request, response, next) => {
  console.log("middleware");
  next();
});

router.route("/orders").get((request, response) => {
  dboperations.GetOrders().then((result) => {
    response.json(result);
  });
});
router.route("/order/:id").get((request, response) => {
```

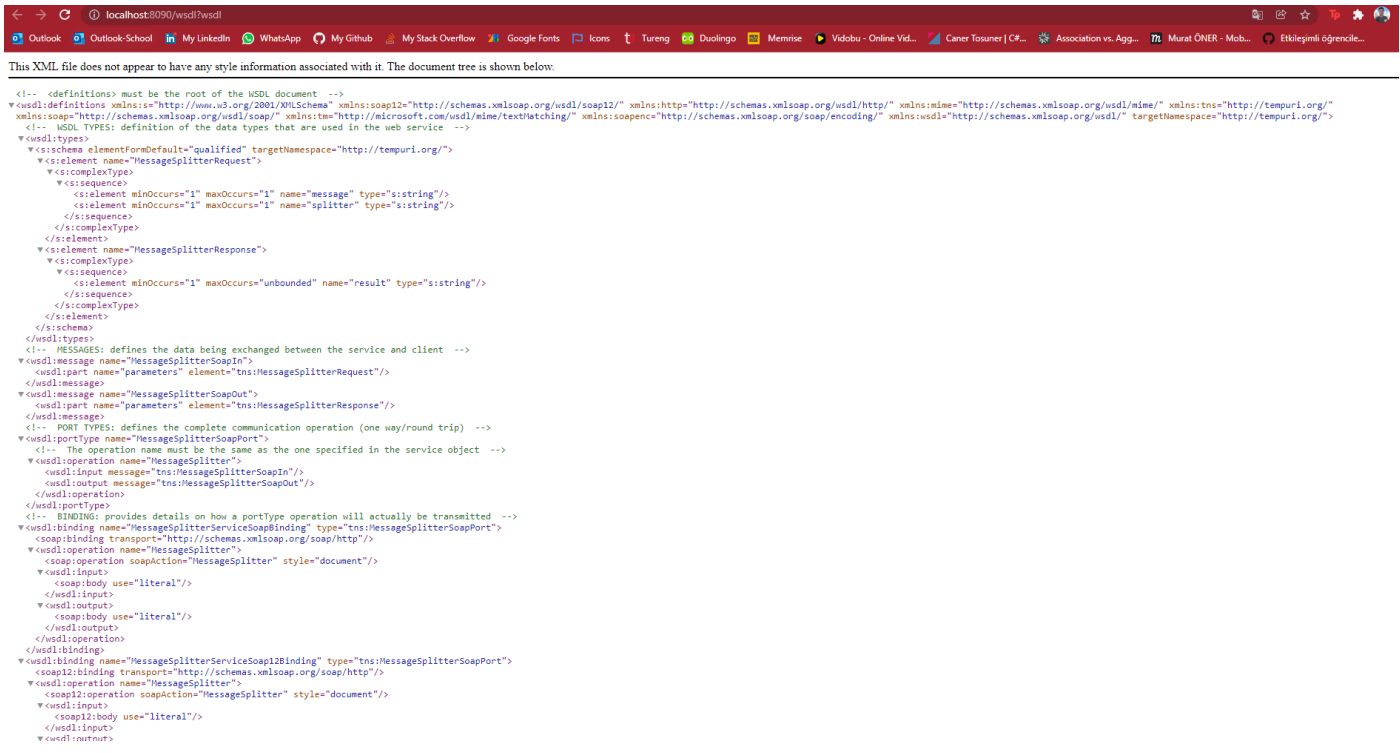
```
  dboperations.GetOrder(request.params.id).then((result) => {
    response.json(result);
  });
});
router.route("/addOrder").post((request, response) => {
  let order = { ...request.body };


  dboperations.AddOrder(order).then((result) => {
    response.status(201).json(result);
  });
});
router.route("/deleteOrder/:id").get((request, response) => {
  dboperations.DeleteOrder(request.params.id).then((result) => {
    response.status(200).json(result);
  });
});
router.route("/updateOrder/:id").post((request, response) => {
  let order = { ...request.body };


  dboperations.UpdateOrder(request.params.id, order).then((result) => {
    response.status(200).json(result);
  });
});
router.route("/").get((request, response) => {
  response.send(
    'Node Soap Example!<br /><a href="https://github.com/macogala/node-soap-example#readme">Git README</a>'
  );
});
```

```
var port = 8090;
app.listen(port, function () {
  console.log("Listening on port " + port);
  var wsdl_path = "/wsdl";
  soap.listen(app, wsdl_path, serviceObject, xml);
  console.log(
    "Check http://localhost:" +
      port +
      wsdl_path +
      "?wsdl to see if the service is working"
  );
});
```

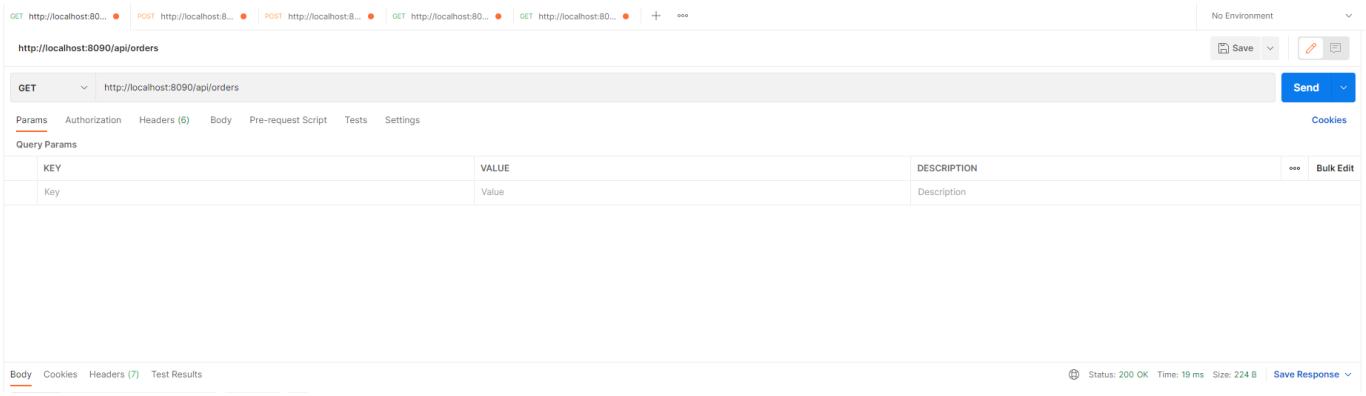
Service.wsdl :



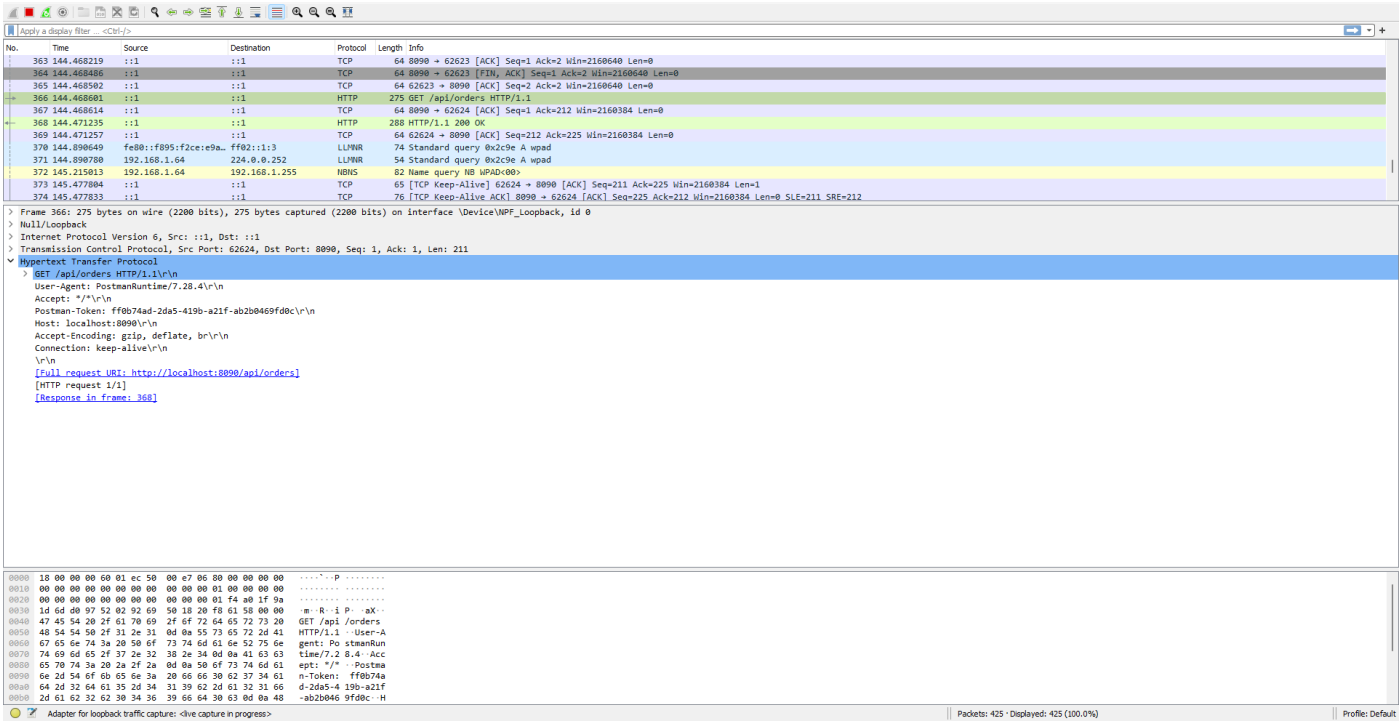
ÇALIŞTIRMA

GetOrders:

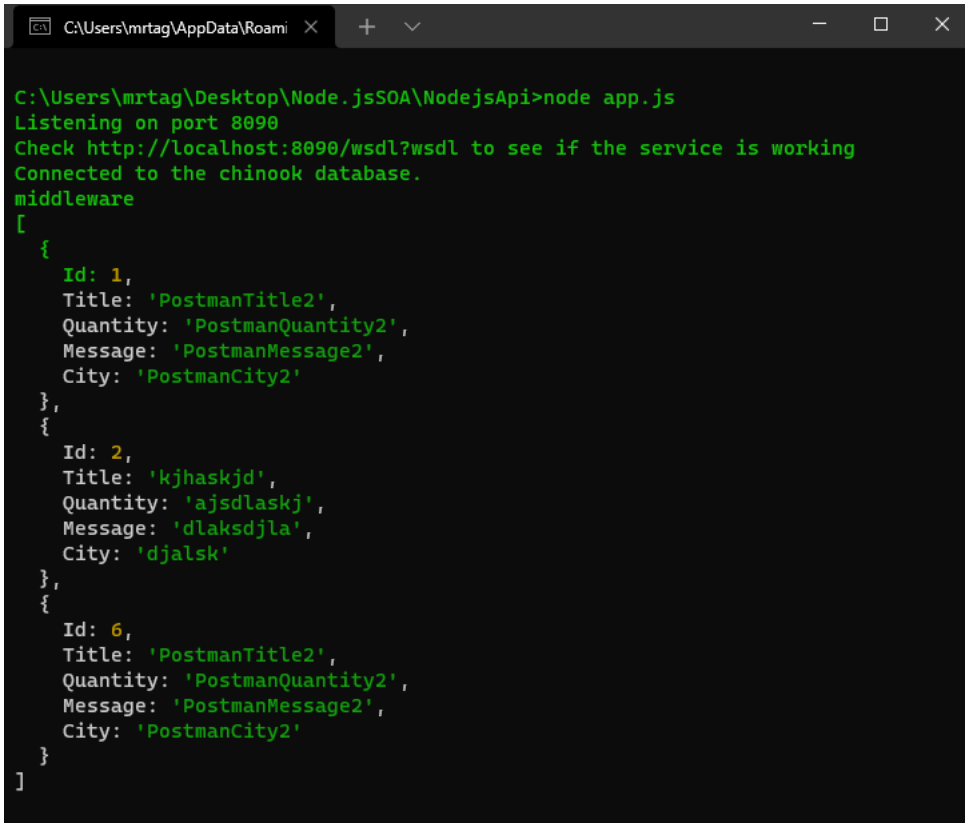
Postman



Wireshark

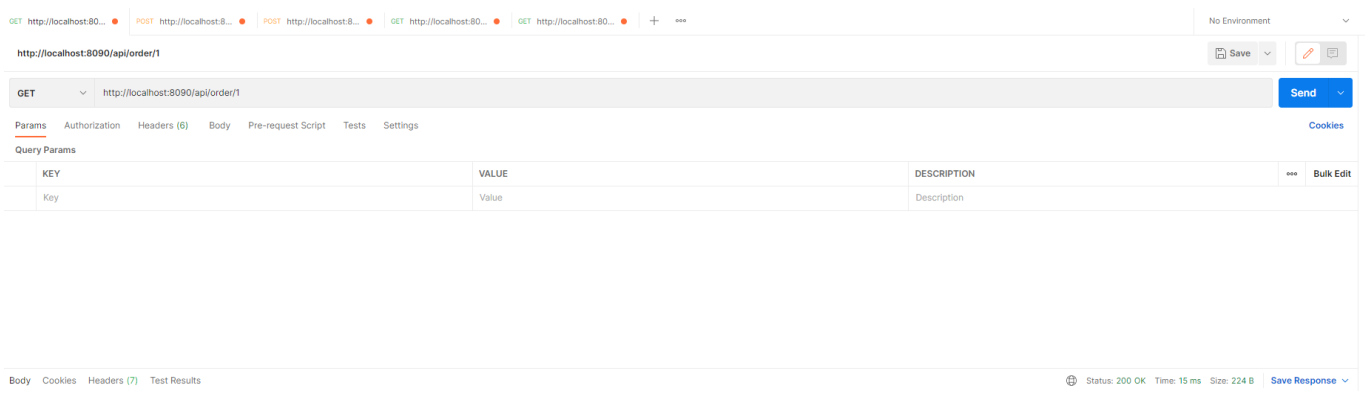


Çıktı

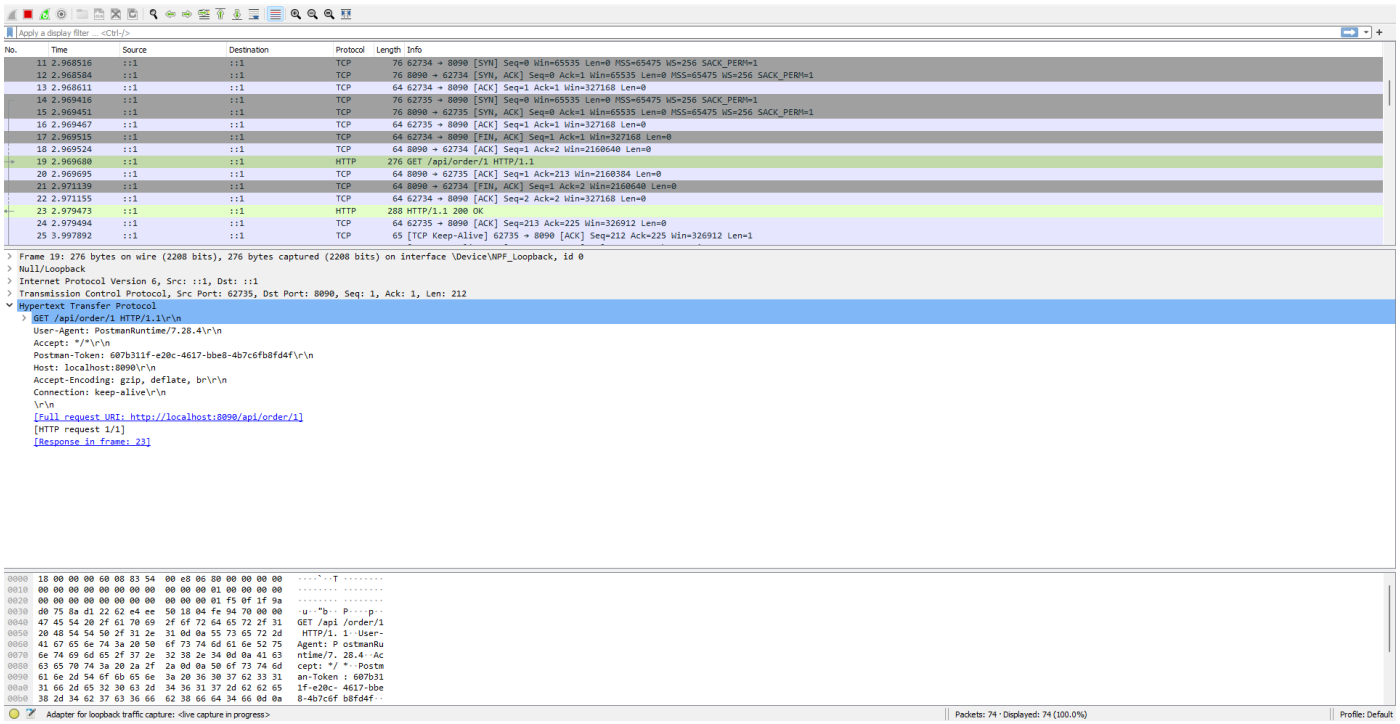


GetOrder:

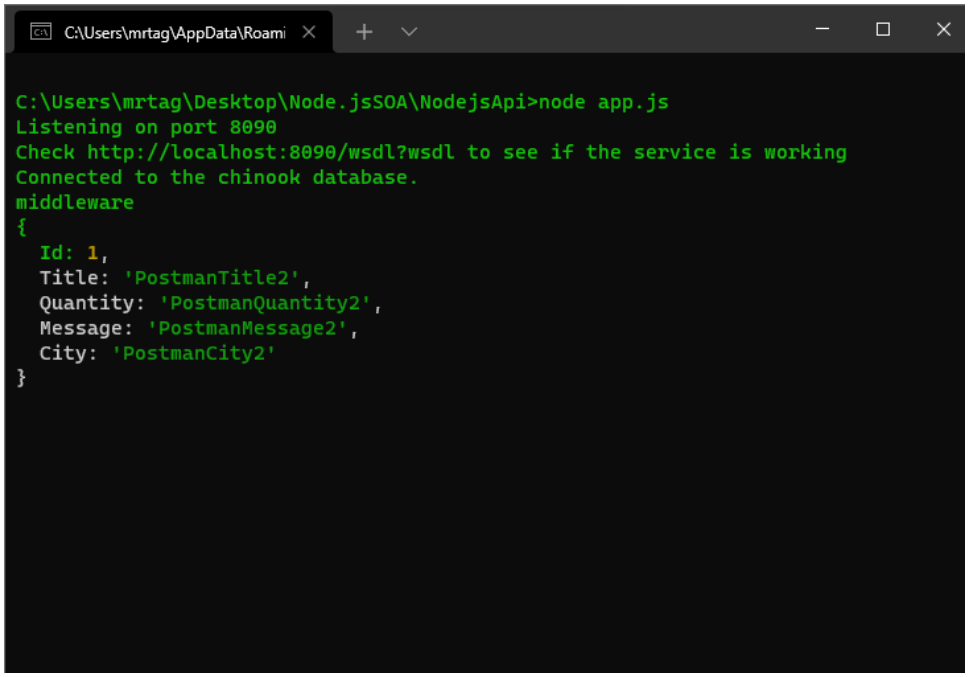
Postman



Wireshark

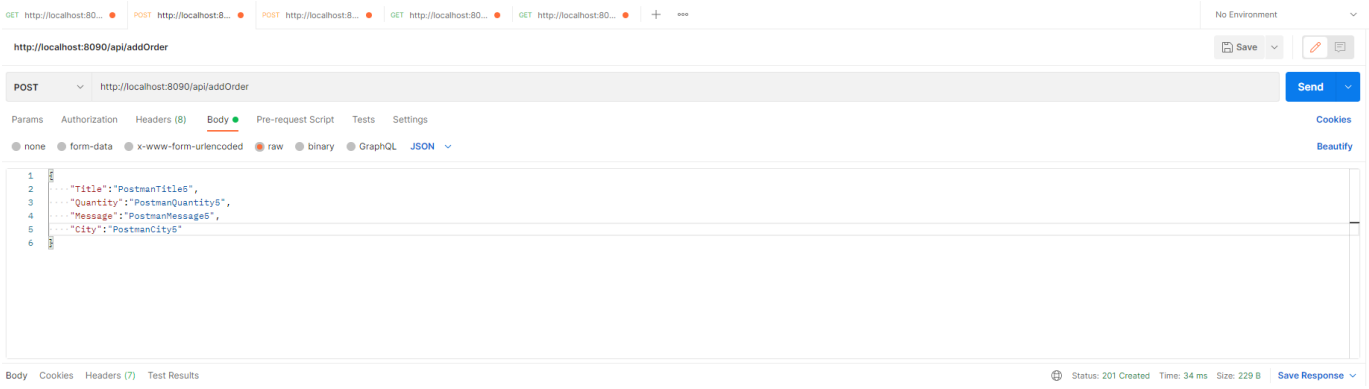


Çıktı

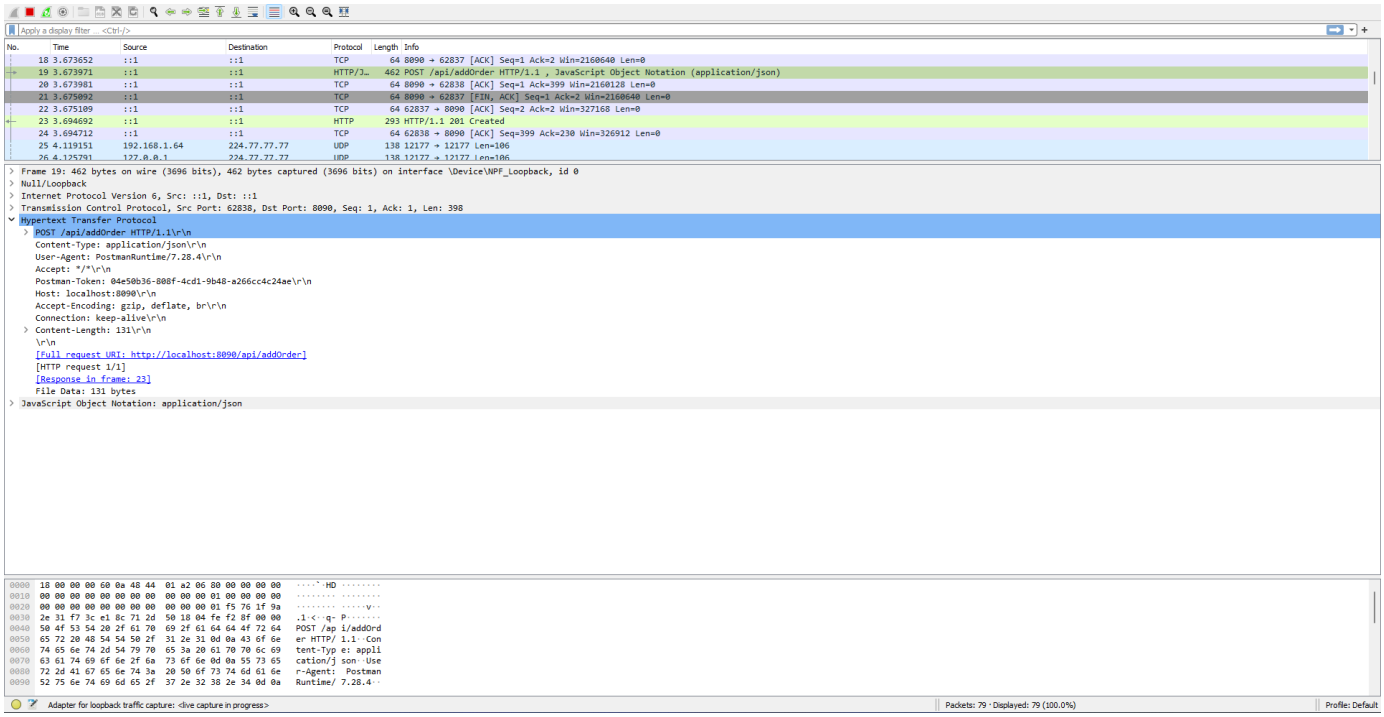


AddOrder:

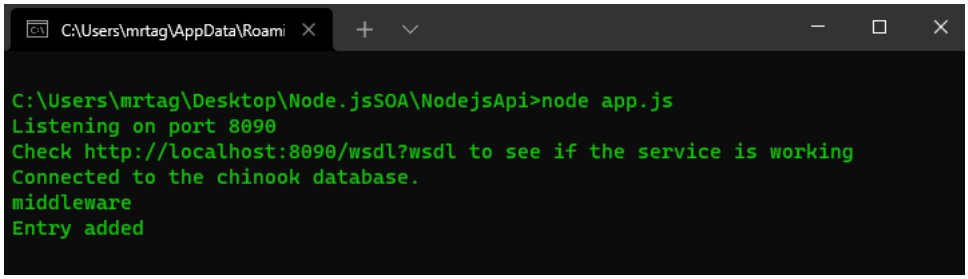
Postman



Wireshark



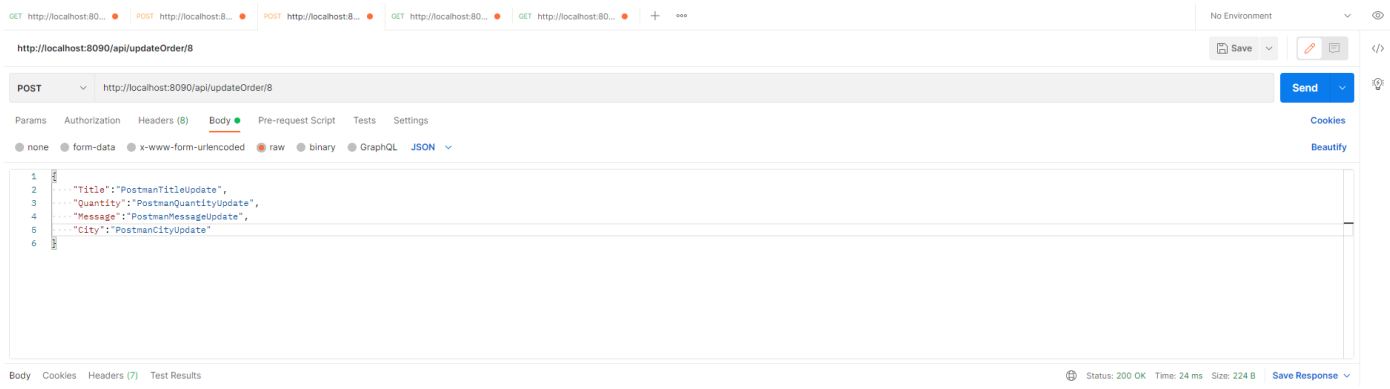
Çıktı



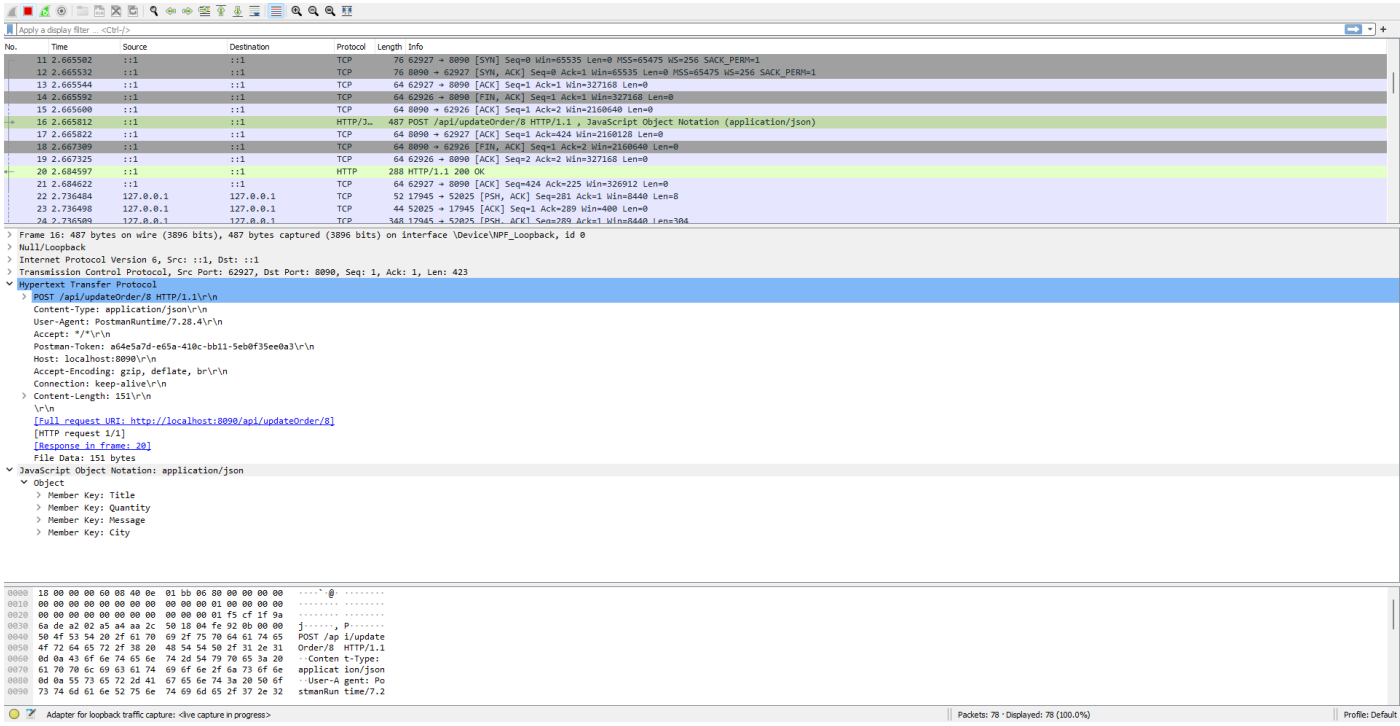
	Id	Title	Quantity	Message	City
	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	PostmanTitle2	PostmanQuantity2	PostmanMessage2	PostmanCity2
2	2	kjhaskjd	ajsdlaskj	dlaksdjla	djalsk
3	6	PostmanTitle2	PostmanQuantity2	PostmanMessage2	PostmanCity2
4	8	PostmanTitle5	PostmanQuantity5	PostmanMessage5	PostmanCity5

UpdateOrder:

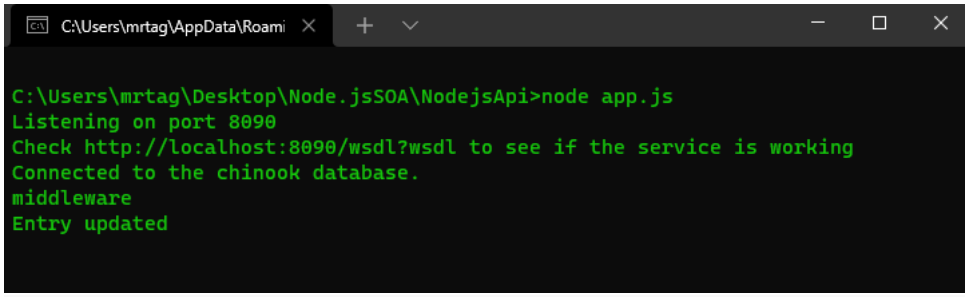
Postman



Wireshark



Çıktı

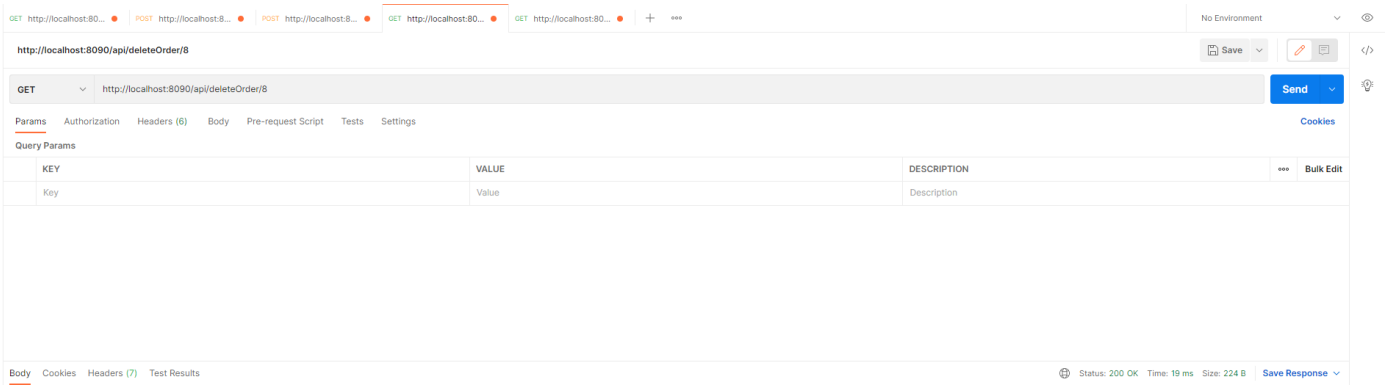


Tablo: Orders — Filter in any column

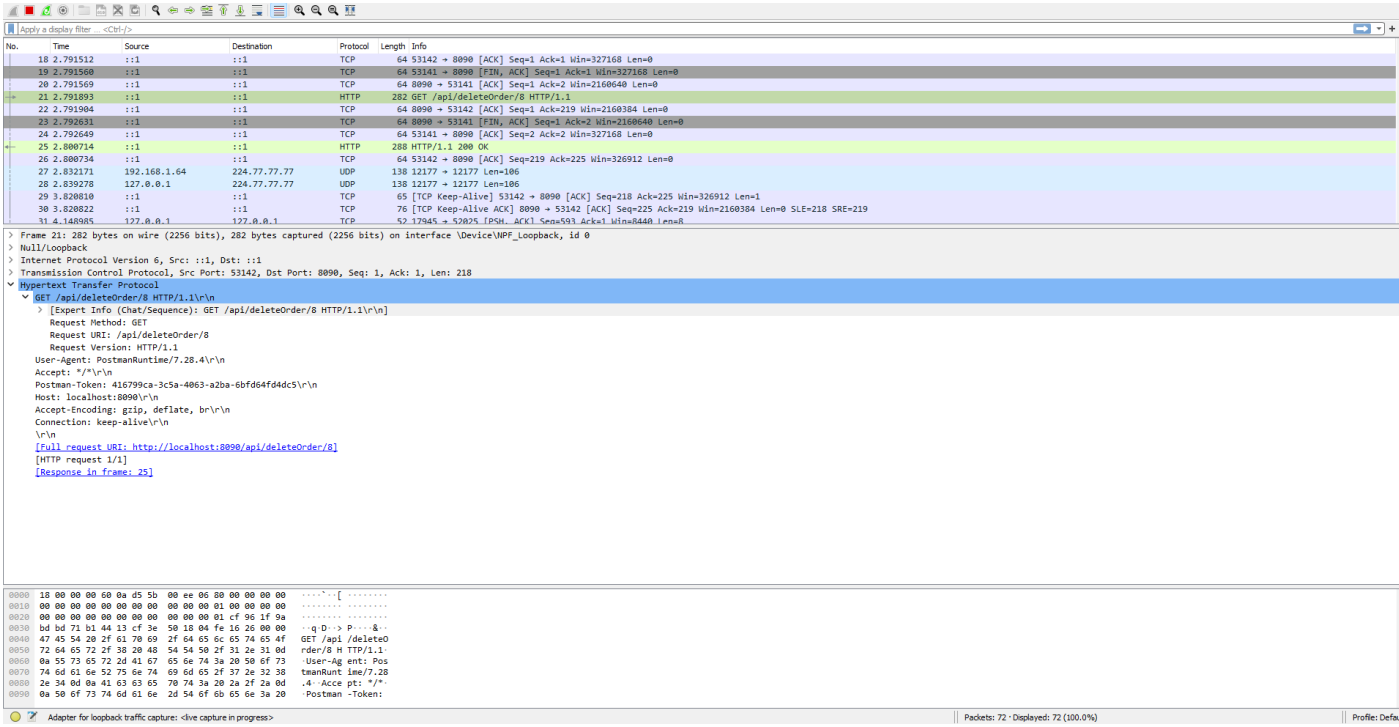
	Id	Title	Quantity	Message	City
	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	PostmanTitle2	PostmanQuantity2	PostmanMessage2	PostmanCity2
2	2	kjhaskjd	ajsdlaskj	dlaksdjla	djalsk
3	6	PostmanTitle2	PostmanQuantity2	PostmanMessage2	PostmanCity2
4	8	PostmanTitleUpdate	PostmanQuantityUpdate	PostmanMessageUpdate	PostmanCityUpdate

DeleteOrder:

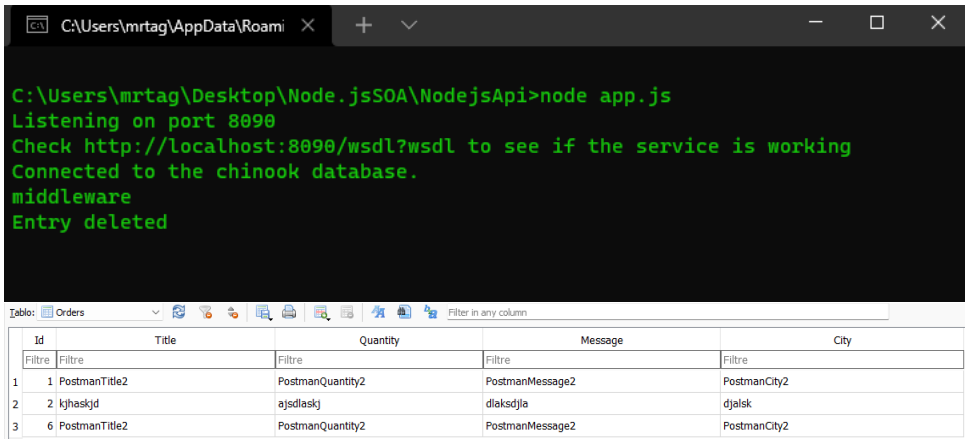
Postman



Wireshark



Çıktı

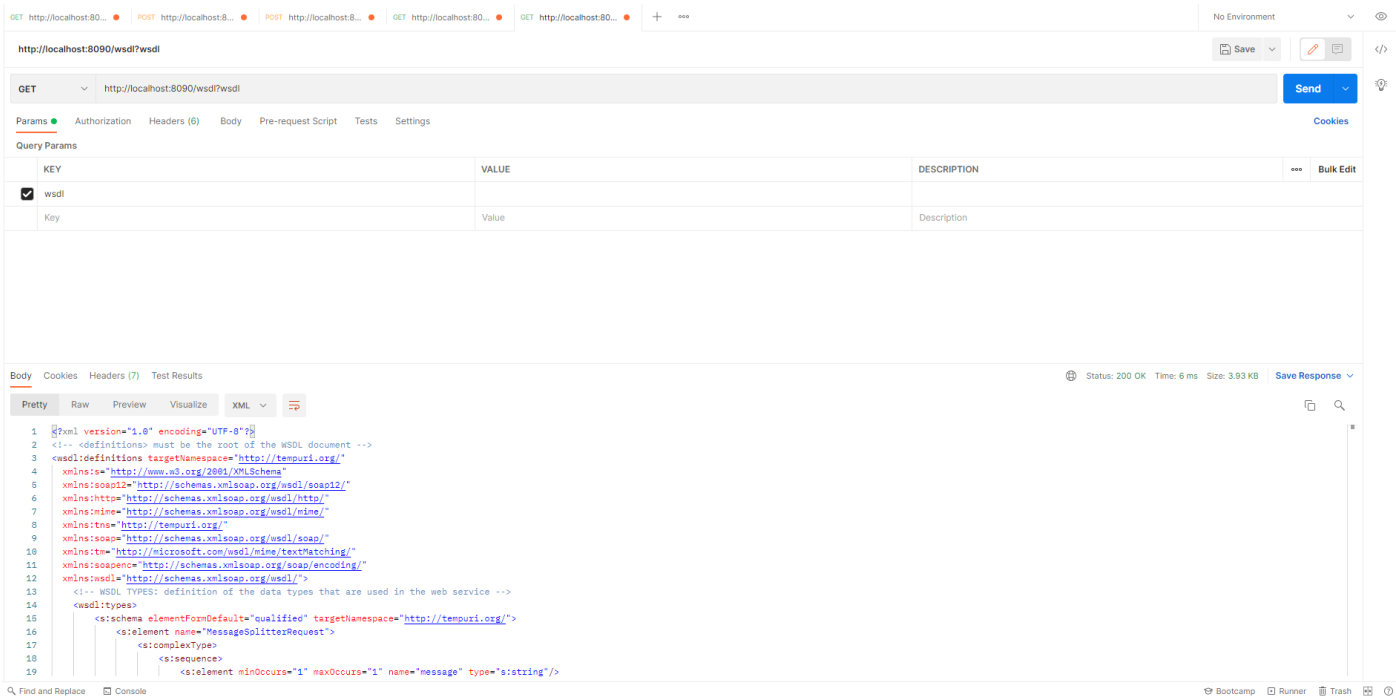


Tablo: Orders — Filter in any column

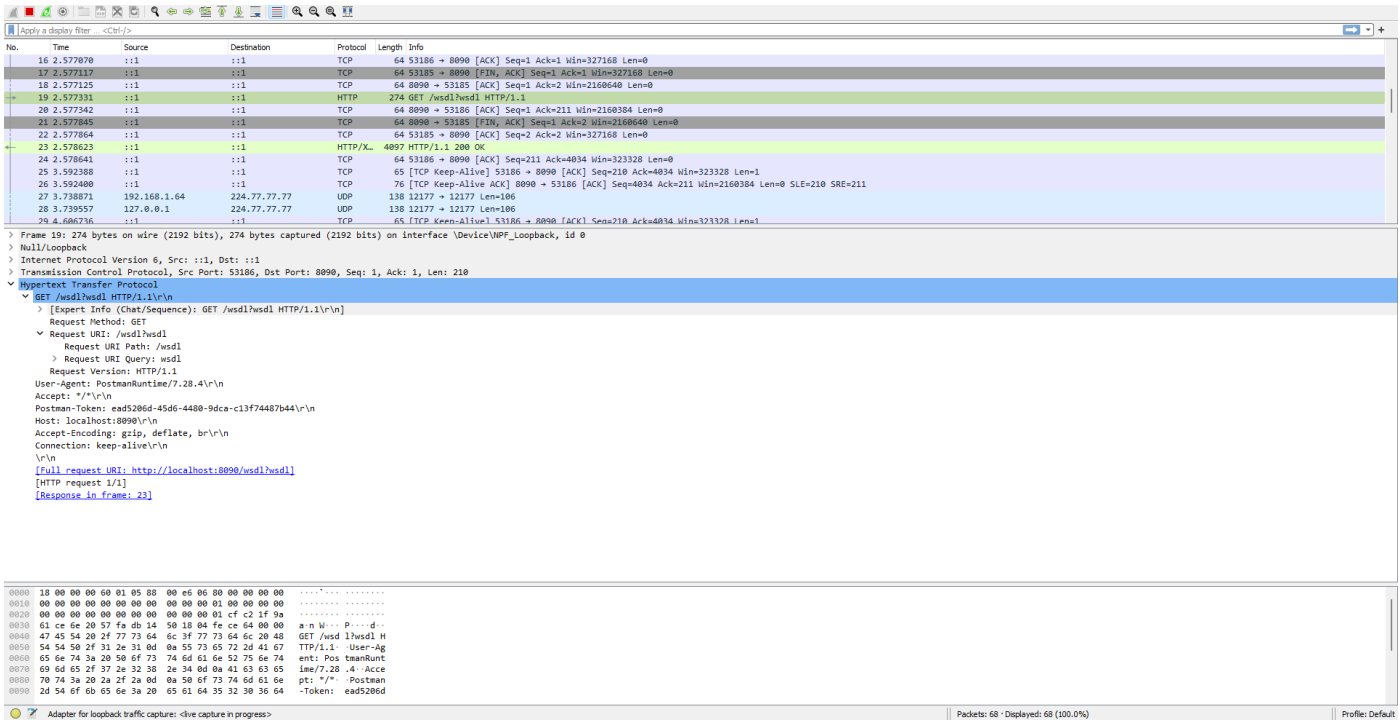
	Id	Title	Quantity	Message	City
	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	PostmanTitle2	PostmanQuantity2	PostmanMessage2	PostmanCity2
2	2	kjhaskjd	ajsdlaskj	dlaksdjla	djalsk
3	6	PostmanTitle2	PostmanQuantity2	PostmanMessage2	PostmanCity2

SOAP/WSDL:

Postman



Wireshark



Çıktı