INHALTSVERZEICHNIS

Gruppe 2: Lucas Gomes, Kevin Schulz, Jakob Prodöhl

SQL-DATENBANK
DOCKER COMPOSE

```
version: '3'
services:
  db:
    image: mysql:5.7
    container_name: wi_g002_mysql
    environment:
      MYSQL_ROOT_PASSWORD: PASSWORD
      MYSQL_DATABASE: wi_g002_shipments_api
      MYSQL_USER: wi_g002
      MYSQL_PASSWORD: PASSWORD
    ports:
      - "XXXX:3306"
    volumes:
      - dbdata:/var/lib/mysql
      - type: bind
          source: ./config
          target: /etc/temp
          consistency: consistent
volumes:
  dbdata:
```

Docker-Image

Umgebungsvariablen für die Verbindung und Konfiguration

Verbindungsport

Speicherkonfigurationen

Gruppe 2: Lucas Gomes, Kevin Schulz, Jakob Prodöhl

SQL-DATENBANK

KONFIGURATION DER DATENBANK

interaktiver Befehl

```
(1)                          docker compose up
```

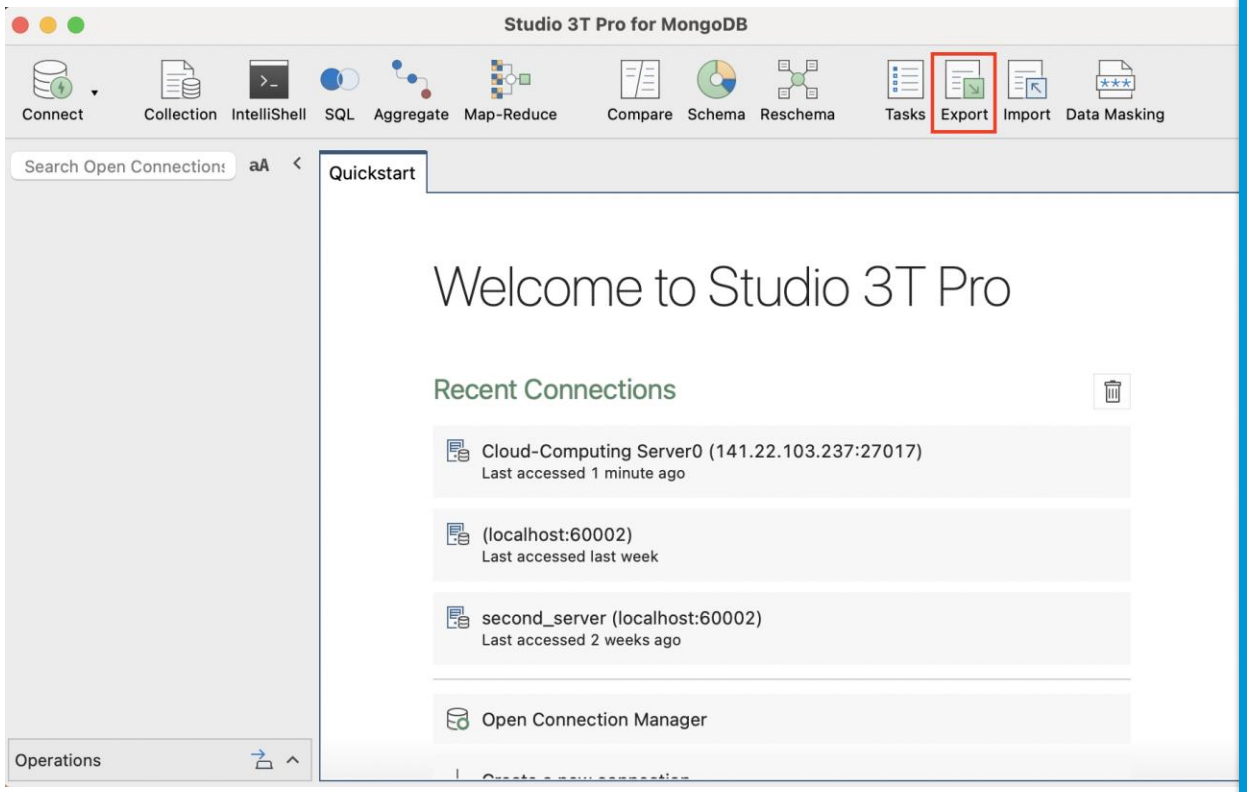
```
(2)                    docker exec -it wi_g002_mysql bash
```

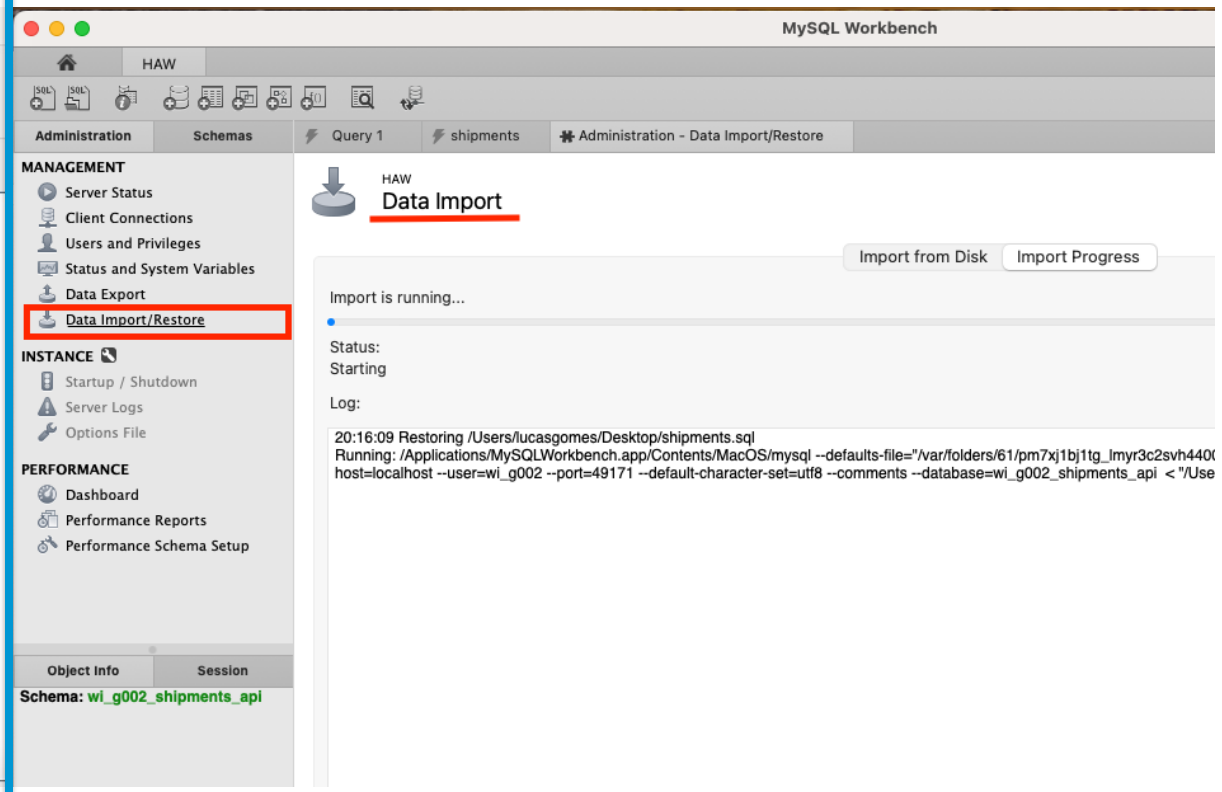
```
(3)                           mysql -uroot -p
```

```
(4)              GRANT ALL PRIVILEGES ON *.* TO 'wi_g002'@'%';
```

```
(5)  ALTER USER 'wi_g002'@'%' IDENTIFIED WITH  mysql_native_password BY 'PASSWORD';
```

SQL-DATENBANK
DATENÜBERTRAGUNG





SQL-DATENBANK
DATENÜBERTRAGUNG

```
(1)                    docker exec -it wi_g002_mysql bash
```

```
(2)     mysql -u root  -p wi_g002_shipments_api < /etc/temp/shipments.sql
```

Datenbank importieren

SQL-DATENBANK

SHIPMENTSAPI -> ./DOCKER

```
# Specifies the base image we're extending
FROM node:12
# Specify the "working directory" for the rest of the Dockerfile
WORKDIR /src
# Install packages using NPM (bundled with the node:12 image)
COPY ./app/package.json /src/package.json
RUN npm install
RUN npm install mysql
# Or list each module install here
# Add application code
COPY ./app/server.js /src/app/server.js
COPY ./app/api /src/app/api
# Set environment to "development" by default
ENV NODE_ENV development
# Run the application directly via node
CMD ["node", "/src/app/server.js","start"]
```

Abhängigkeiten anpassen

SQL-DATENBANK

SHIPMENTSAPI-> ./APP/API/**DATABASECONFIG.JS**

```
var mysql = require('mysql');

config = {
  host     :  process.env.ADRDB || '141.22.103.237',
  user     : 'wi_g002',
  password : 'MyNewPass',
  database : 'wi_g002_shipments_api',
  port: process.env.DBPORT || 6033
}
var connection =mysql.createConnection(config);  //added the line
connection.connect(function(err){
  if (err){
    console.log('error connecting:' + err.stack);
  }
  console.log('connected successfully to DB.');
});

module.exports ={
     connection : mysql.createConnection(config)
}
```

Abhängigkeiten anpassen

Feedback

Exportvariablen

SQL-DATENBANK

SHIPMENTSAPI-> ./APP/API/CONTROLLER/**SHIPMENTSCONTROLLER.JS**

```
'use strict';
//var mongoose = require('mongoose'),
//Shipment = mongoose.model('Shipments');
var Server1='http://141.22.103.237';
var port = process.env.CPORT || 3002;

var mysql      = require('mysql');
var config = require('../databaseConfig.js');
var connection= config.connection
connection.connect();

connection.on('error', function(err) {
    console.log(err);
});

[…]
connection.query('SELECT * FROM wi_g002_shipments_api.shipments where PLZ_From like "'
                        +req.params.plz+'%";', function (err, Shipment, fields)
[…]
```

Variablen importieren

Befehlsanpassung

Gruppe 2: Lucas Gomes, Kevin Schulz, Jakob Prodöhl

SQL-DATENBANK

BEFEHL

Behandlung von Verbindungsfehlern

```
docker run --restart on-failure -p 3012:3000 -d -e 'CPORT=3012' -e 'DBPORT=6033'
 -e 'ADRDB=141.22.103.237' --network wi_g002_bridge --name wi_g002_shipmentsapi
                           wi_g002_shipmentsapi_X
```

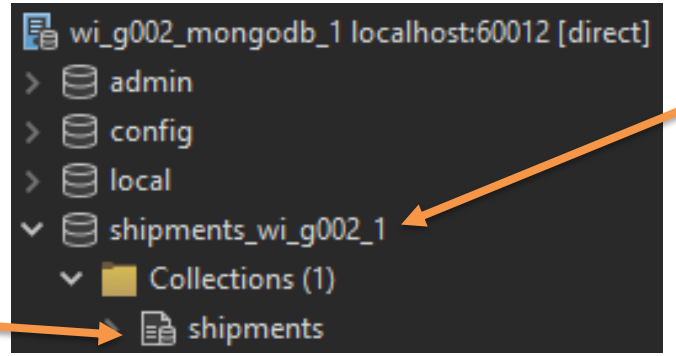
1. Anpassen des Datenbankpfads der ShipmentAPI

server.js

```
mongoose.connect('mongodb://wi_g002_mongodb_1:27017/shipments_wi_g002_1');
```

Name des Datenbank-Containers

Name der Datenbank innerhalb des Containers



(Liefer-)Daten

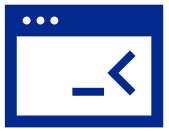
2. Bauen des docker images

```
docker build -t wi_g002_shipmentsapi_1 .
```

3. Starten des ShipmentsAPI-Containers

Einzigartiger Port für den Container

Übergabe eines zusätzlichen Parameters



```
docker run -p 3012:3000 -e 'CPORT=3012' --network wi_g002_bridge
--name wi_g002_shipmentsapi_1 wi_g002_shipmentsapi_1
```

4. Starten des mongoDB-Containers

Dateipfad für persistente Daten

```
docker run --rm -p 60012:27017 -v ~/mongodbdata_1/data:/data/db
--network wi_g002_bridge --name wi_g002_mongodb_1 -d mongo:3.6
```

Name wie in server.js angegeben

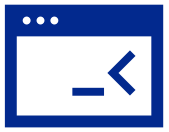
5. Datenbank mit Daten beladen

Datenbank wie in server.js angegeben

```
docker cp shipments.txt wi_g002_mongodb_1:shipments.txt
```

```
docker exec -it wi_g002_mongodb_1 mongoimport -d shipments_wi_g002_1 -c
shipments --type TSV --file shipments.txt --headerline -columnsHaveTypes
```

6. Neue ShipmentsAPI-Container in Loadbalancer aufnehmen

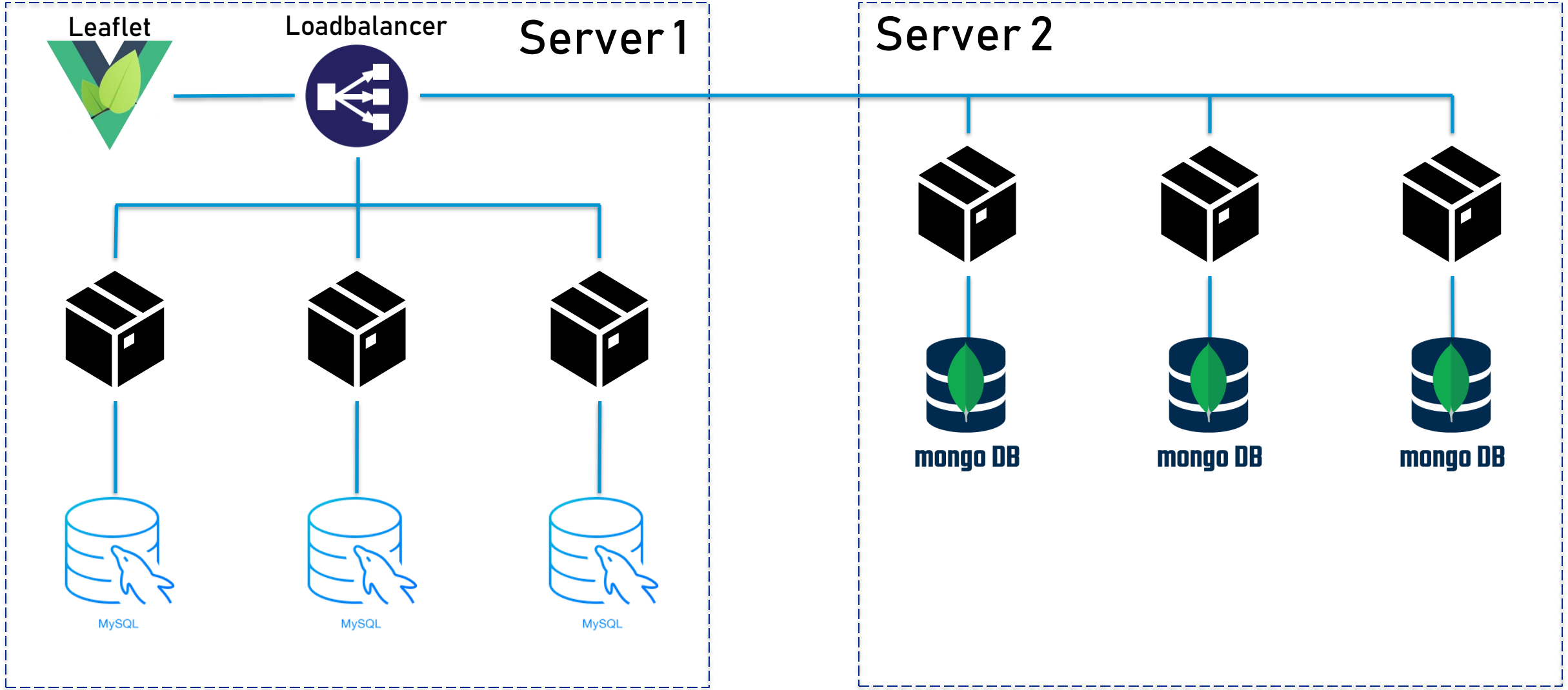


```
docker run --rm -p 8002:8080 -e 'SERVERS=http://141.22.102.166:3012
http://141.22.102.166:3022 http://141.22.102.166:3032'
-e 'LB_STRATEGY=RR' --name wi_g002_loadbalancer_so wi_g002_loadbalancer_so
```

IP:Port der erstellten ShipmentsAPI-Container

7. Leaflet-Container starten und an den Loadbalancer anbinden

NETZWERKAUFBAU



Gruppe 2: Lucas Gomes, Kevin Schulz, Jakob Prodöhl
Abschlusspräsentation Cloud Computing

ShipmentsAPI-Container

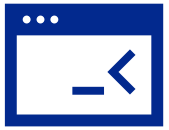
HAW HAMBURG

API-REQUESTS PER CONTAINER

1. ANSATZ

Notwendige Informationen console.log() in  lb.js

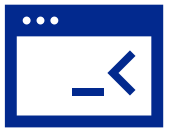
```
new cur: 1
Completed on  http://141.22.103.237:3012 GET /shipments/plz
```



```
console.log('Completed on ', originalHost, req.method, req.url, Date.now() -
start);
```

Request und Response werden durch den Loadbalancer nur weitergeleitet

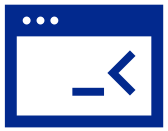
--> über shipmentsRoutes.js durch Funktion in shipmentsController.js abgearbeitet



```
const originalHost = 'http://'+req.readableState.pipes.originalHost;
```

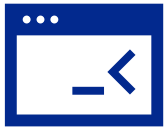
API-REQUESTS PER CONTAINER
UMSETZUNG

shipmentsController.js



```
var Server1='http://141.22.103.237';
var port = process.env.CPORT || 3002;
```

Für alle list_shipments-Funktionen



```
var infoJSON = JSON.stringify(Server1 + req.url + ' over Port ' + port);
Shipment.push(infoJSON);
```

Bei Erstellung der Shipments-API:

zusätzlich den Port des Containers mit -e 'CPORT=30XX‘ angeben

API-REQUESTS PER CONTAINER

VERARBEITUNG & VISUALISIERUNG

app.js

```
Success: function(data) {
        let popped = data.pop();
    info.update(e.target.feature.properties, aggregateData(data), PackageNo(data));
    info2.update(e.target.feature.properties, aggregateRequest(popped), popped, servers);
```

API-REQUESTS PER CONTAINER

VISUALISIERUNG

Demonstration der Umsetzung live in Leaflet

ZUSATZ
ANZAHL DER PAKETE

app.js

```
function PackageNo(data) {
    var ShipmentId=[];
    data.forEach(item=> ShipmentId.push(item.ShipmentId) );
    return ShipmentId.length;}
```

ZUSATZ

NEUSTE SHIPMENT-IDS

app.js

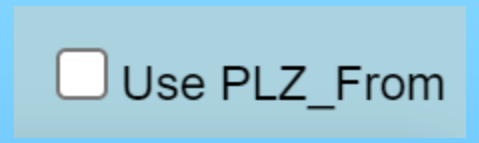
```
function ShipID(data) {
    let text = "";
    var ShipmentId=[];
    data.forEach(item=> ShipmentId.push(item.ShipmentId) );
    var stopreq = ShipmentId.length/5;
    stopreq = Math.min(stopreq,5);
    for (let i = 0; i < stopreq ; i++) {
        text += ShipmentId.splice(ShipmentId.length-5, ShipmentId.length) + "<br>";}
    return text;}
```

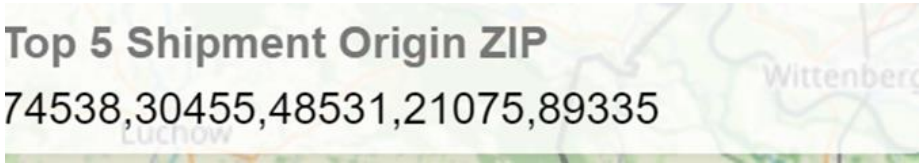
ZUSATZ

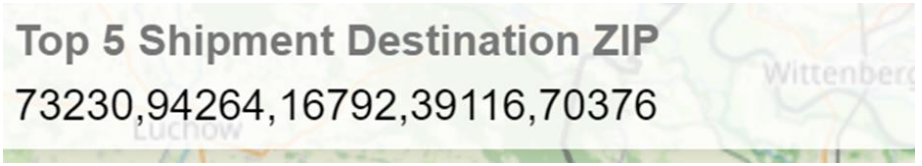
TOP PLZ + RICHTUNG

app.js

```
function topPLZ(data) {
    let text = "";  var topplz=[];  var allplz=[];
    if (direction === "from"){
        data.forEach(item=> allplz.push(item.PLZ_To) );
        topplz = [...new Set(allplz)];
        topplz = topplz.slice(0, 5);
        text = '<p><b>Top 5 Shipment Destination ZIP </b></p>' + topplz;}
    else{  data.forEach(item=> allplz.push(item.PLZ_From) );
        topplz = [...new Set(allplz)];
        topplz = topplz.slice(0, 5);
        text = '<p><b>Top 5 Shipment Origin ZIP </b></p>' + topplz;}
    return text;}
```







VIELEN DANK FÜR DIE AUFMERKSAMKEIT

HAW HAMBURG

API-REQUESTS PER CONTAINER
FUNKTION

app.js

```
function aggregateRequest(popped) {
    let text = "";
    var requestcount=[];
    ip=popped.slice(1,22)+':'+popped.slice(popped.length-5,popped.length-1);
    requestlist.push(ip);
    console.log(requestlist);
    servers = [...new Set(requestlist)];
    console.log(servers);
    for (let k = 0; k <= servers.length; k++) {
        requestcount[k]=0;
        requestlist.forEach(element => {
            if (element === servers[k]) {
            requestcount[k] += 1;}});}
    console.log(requestcount);
```

API-REQUESTS PER CONTAINER

FUNKTION & VISUALISIERUNG

app.js

```
//Erstellen der Textausgabe
for (let i = 1; i <= servers.length; i++) {
    text += 'Container' + i + ' ' + servers[i-1].slice(0,21) + ' Port ,
                + servers[i-1].slice(22,servers[i-1].length)+ ': ' + requestcount[i-1]
                + "<br>";}
return text;}
```

```
info2.update = function (props, apiinfo, currentrequest, servers) {
    this._div.innerHTML =      (props ? '<p><b>Current request: </b></p>` +
            currentrequest + '<br/><br/>`
        + '<p><b>Total number of api requests per Container </b></p>' + apiinfo
                : '<p><b>API Information </b></p>' + 'Hover over a district');};
```

ZUSATZ
RICHTUNG

app.js

```
//Change if PLZ_To or PLZ_From is used
var directionControl = new L.control({position:'topright‘});
directionControl.onAdd = function (map) {
    var div = L.DomUtil.create('div', 'command‘);
    div.innerHTML = '<form><input type="checkbox" id="myCheckbox"/>
                    Use PLZ_From</form>';
    return div;
    console.log(div);};
directionControl.addTo(map);
```

```
const checkbox = document.getElementById('myCheckbox‘)
checkbox.addEventListener('change', (event) => {
    if (event.currentTarget.checked) {
        console.log("Change direction");
        direction ="from";  }
    else {
        console.log("Change direction");
        direction ="to";  }})
```

SQL-DATENBANK

SHIPMENTSAPI-> ./APP/**SERVER.JS**

```
var express = require('express'),
app = express(),

port = process.env.PORT || 3000,
cport = process.env.CPORT || 3002,

bodyParser = require('body-parser');

process.on('uncaughtException', err => {
  console.error('There was an uncaught error', err);
  process.exit(1);
});

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
var routes = require('./api/routes/shipmentsRoutes'); //importing route
routes(app); //register the route
app.listen(port);
console.log('ok')
console.log('shipment RESTful API server started on: ' + cport);
```

Fehlerbehandlung