Robert Baumgartner

Verlagere die Daten, die sich bisher in einem JSON File befanden, in die PostgreSQL Datenbank **cars**. Dazu habe ich dir die Datei **cars.sql** erstellt. Schau dir diese Datei im Editor an. Speziell **cars_id_seq.**

Wie viele Einträge gibt es bei **cars** und womit beginnt die Sequenz **cars_id_seq**? Was wird also passieren, wenn du neue Autos mit INSERT in die Tabelle einfügst, ohne eine **id** zu vergeben? Eben. Also modifiziere die Datei **cars.sql** vor dem Import! Übrigens kannst du mit:

```
SELECT nextval('cars_id_seq');
```

den Sequenzwert um den Incrementwert erhöhen und dich mit

```
SELECT currval('cars_id_seq');
```

über den aktuellen Wert informieren. BTW alle Sequenzen haben immer den Namen ***tabelle_spalte*_seq**.
Lege einen speziellen User an, beschränke die Zugriffsmöglichkeit!

Die Datei **cars.js** im verzeichnis **routes** ist vorgegeben:

```
const express = require('express');
const { getCars, deleteCar, changeStatusCar, addCar } = require('../controllers/cars');

const router = express.Router();

router.get('/cars', getCars);
router.patch('/cars/:id', changeStatusCar);
router.delete('/cars/:id', deleteCar);
router.post('/cars', addCar);

module.exports = router;
```

Im Verzeichnis **controllers** befinden sich die Funktionen für die Routen (Datei **cars.js**).

Im Verzeichnis **model** befinden sich die Funktionen um mit der Datenbank zu interagieren (Datei **cars.js**).

Die Bilder befinden sich **images.zip**.

Wenn du frisch beginnen willst und nicht deinen bereits vorhandenen Code verwenden möchtest, gibt es die Datei **Server Angabe.zip.**

Daten fürs Einfügen:

```
{
  "title": "Rare Oldtimer",
  "status": "available",
  "price": "52300",
  "image": "http://localhost:3000/images/car10.png",
  "miles": 189921,
  "yearOfMake": 1978,
  "owner": {
    "firstName": "Max",
    "lastName": "Blam"
  },
  "description": "Excellent condition. A trip back in time!"
}
```

Robert Baumgartner

HTL WIEN WEST

Erstelle den REST Client und teste deine Routen.

- Alle Routen implantiert wie angegeben
- Spezieller User mit eingeschränkten Aktionen verwendet
- Code aud Model, Controller, Routes Verzeichnisse aufgeteilt
- AirBnB Styleguide einhalten
- Error Handlung implementiert
- ES Lint fehlerfreier Code erstellt

Beispiele:

```
DELETE http://localhost:3000/carsssss/2
```

```
> nodemon app.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
Not found ====> /carsssss/2
DELETE /carsssss/2 404 4.939 ms - 66
```

```
DELETE http://localhost:3000/cars/2
```

```
11  X-XSS-Protection: 0
12  Content-Type: text/html; charset=utf-8
13  Content-Length: 7
14  ETag: W/"7-RBvabNhWieR26+EEQPJ5Z/rvYaY"
15  Date: Mon, 20 Sep 2021 09:10:21 GMT
16  Connection: close
17
18  Deleted
```

```
DELETE http://localhost:3000/cars/22
```

```
12  Content-Type: text/html; charset=utf-8
13  Content-Length: 35
14  ETag: W/"23-scUjbYnn/DCnzHk6r4vBqyyXaIo"
15  Date: Mon, 20 Sep 2021 09:11:35 GMT
16  Connection: close
17
18  id 22 was not found in the database
```