Universidad Tecnológica de Xicotepec de Juárez

Consumo de API - Automóviles

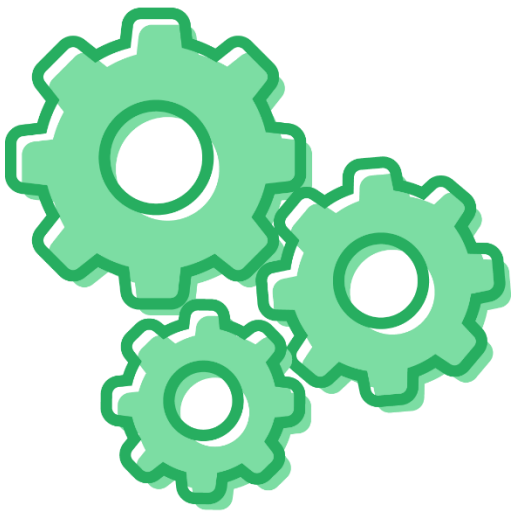
Carlos Martin Hernández de Jesús – 210496

9*A IGDGS

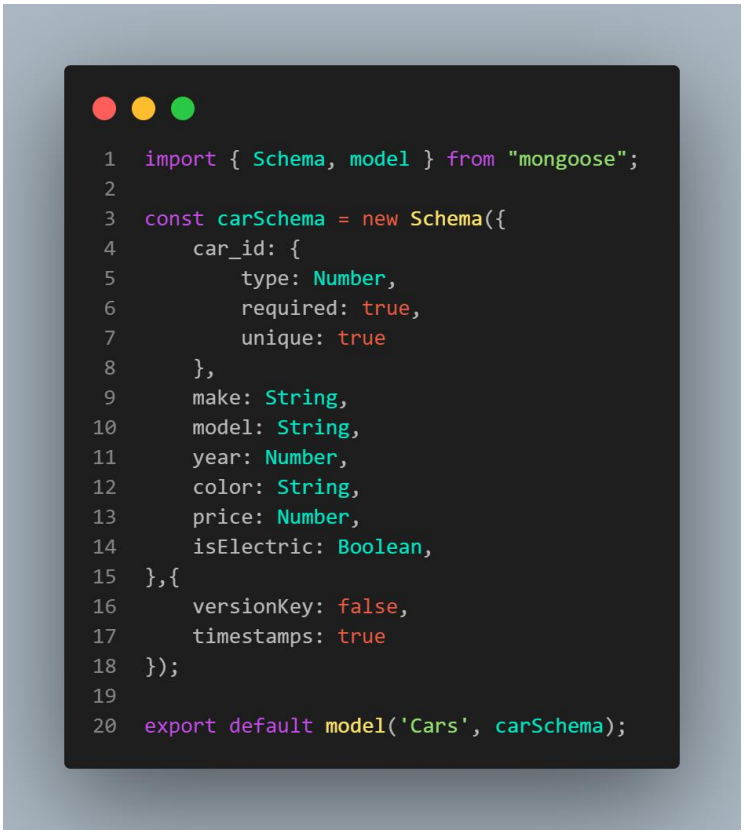
Profesor:

Luna Trejo Cupertino

martes, 13 de agosto de 2024



MODELO - Automóviles



```
import { Schema, model } from "mongoose";

const carSchema = new Schema({
    car_id: {
        type: Number,
        required: true,
        unique: true
    },
    make: String,
    model: String,
    year: Number,
    color: String,
    price: Number,
    isElectric: Boolean,
},{
    versionKey: false,
    timestamps: true
});

export default model('Cars', carSchema);
```

El modelo contiene los elementos:

- Car_id – Identificador.
- Make – Fabricante.
- Model - Modelo de un automóvil.
- Year – Año en que se fabricó o está inspirado.
- Color – Color del automóvil.
- Price – Precio del automóvil.
- isEelectric – Si es eléctrico o no

PROCEDIMIENTO - getAllCars

DAO – CONTROLLER: Método para obtener todos los carros de la base de datos y controlador para obtener todos los carros, si la operación es exitosa devuelve todos los carros, en caso de que no regresa un mensaje de error.

```
import Cars from "../models/Cars.js";


const carsDao= {};
    carsDao.getAllCars= async()=>{
    return await Cars.find();
};

```

```
import carsDao from "../daos/cars.dao.js";

export const getAllCars = (req,res) => {
    carsDao.getAllCars()
        .then(cars => {
            res.json(cars);
        })
        .catch(err => {
            res.json({ message: err });
        });
};
```

ROUTER - Ruta para obtener todos los carros.

```
router.get("/getAllCars", getAllCars);
```

RESPUESTA:

```
// 20240813205401
// http://localhost:3000/getAllCars

[
  {
    "_id": "66bbe4a3837d98c8d93912f8",
    "car_id": 10465,
    "make": "Porshe",
    "model": "919 Hybrid",
    "year": 2015,
    "color": "Metalic Blue",
    "price": 100,
    "isElectric": false,
    "createdAt": "2024-08-13T22:56:35.813Z",
    "updatedAt": "2024-08-13T22:56:35.813Z"
  },
  {
    "_id": "66bbe567837d98c8d93912fd",
    "car_id": 10466,
    "make": "Ferrari",
    "model": "488 GTB",
    "year": 2018,
    "color": "Rosso Corsa",
    "price": 250000,
    "isElectric": false,
    "createdAt": "2024-08-13T22:59:51.272Z",
    "updatedAt": "2024-08-13T22:59:51.272Z"
  },
  {
    "_id": "66bbe570837d98c8d93912ff",
    "car_id": 10467,
    "make": "Lamborghini",
    "model": "Huracán EVO",
    "year": 2020,
    "color": "Verde Mantis",
    "price": 300000,
```

Para salir de la pantalla completa, pulsa F11

PROCEDIMIENTO - insertCars

DAO – CONTROLLER: Método para insertar un nuevo carro en la base de datos y controlador para insertar un nuevo carro usando los datos del cuerpo de la solicitud, en caso de que no regresa un mensaje de error.

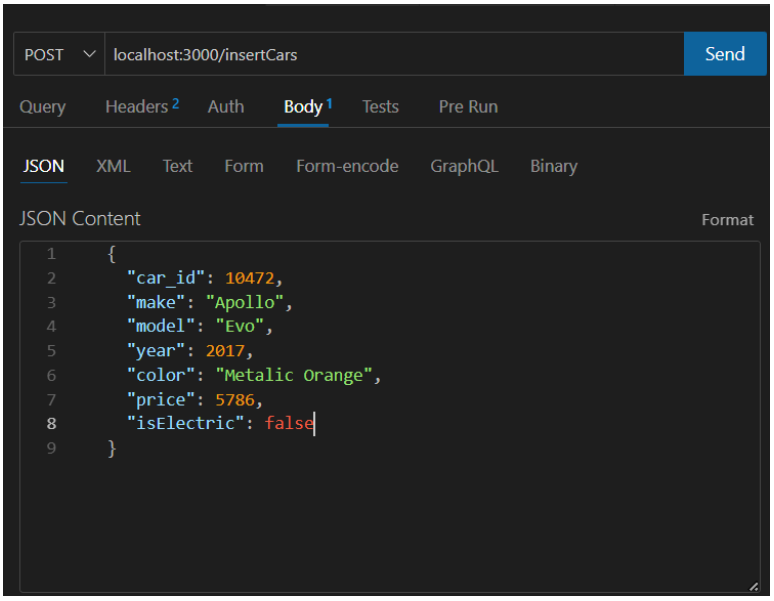
```
carsDao.insertCars= async(cars)=>{
    return await Cars.create(cars);
};
```

```
export const insertCars = (req, res) => {
    carsDao.insertCars(req.body)
        .then(cars => {
            res.json(cars);
        })
        .catch(err => {
            res.json({ message: err });
        });
};
```

ROUTER - Ruta para insertar un nuevo carro.

```
router.post("/insertCars", insertCars);
```

RESPUESTA:



PROCEDIMIENTO - getOneCars

DAO – CONTROLLER: Método para obtener un carro específico basado en su 'car_id' y para obtener un carro específico por su 'car_id', en caso de que no regresa un mensaje de error.

```
carsDao.getOneCars= async(id)=>{
    return await Cars.findOne({car_id:id});
};
```

```
export const getOneCars = (req, res) => {
    carsDao.getOneCars(req.params.id)
        .then(cars => {
            if (cars!=null)
                res.json(cars);
            else
                res.json({ message: "Carrro no Encontrado" });
        })
        .catch(err => {
            res.json({ message: err });
        });
};
```

ROUTER - Ruta para obtener todos los carros por su id.

```
router.get("/getOneCars/:id", getOneCars);
```

RESPUESTA:

```
// 20240813213051
// http://localhost:3000/getOneCars/10465

{
  "_id": "66bbe4a3837d98c8d93912f8",
  "car_id": 10465,
  "make": "Porshe",
  "model": "919 Hybrid",
  "year": 2015,
  "color": "Metalic Blue",
  "price": 100,
  "isElectric": false,
  "createdAt": "2024-08-13T22:56:35.813Z",
  "updatedAt": "2024-08-13T22:56:35.813Z"
}
```

PROCEDIMIENTO - updateOneCar

DAO – CONTROLLER: Método para actualizar un carro específico basado en su 'car_id' y controlador para actualizar un carro específico usando los datos del cuerpo de la solicitud, en caso de que no regresa un mensaje de error.

```
carsDao.findOneUpdate = async (id, updateData) => {
    return await Cars.findOneAndUpdate(
        { car_id: id },
        updateData,
        { new: true }
    );
};
```

```
export const updateOneCar = (req, res) => {
    carsDao.findOneUpdate(req.params.id, req.body)
        .then(car => {
            if (car)
                res.json(car);
            else
                res.json({ message: "Carro no encontrado para actualizar" });
        })
        .catch(err => {
            res.json({ message: err });
        });
};
```

ROUTER - Ruta para actualizar un carro específico por su 'car_id'.

```
router.put("/updateOneCar/:id", updateOneCar);
```

RESPUESTA:

```
    "_id": "66bbe5a4837d98c8d9391307",
    "car_id": 10471,
    "make": "Koenigsegg",
    "model": "Jesko",
    "year": 2024,
    "color": "Papaya Orange",
    "price": 3000000,
    "isElectric": false,
    "createdAt": "2024-08-13T23:00:52.651Z",
    "updatedAt": "2024-08-13T23:00:52.651Z"
}
```

Status: 200 OK Size: 226 Bytes Time: 140 ms

Response Headers 6 Cookies Results Docs

```
{
  "_id": "66bbe5a4837d98c8d9391307",
  "car_id": 10472,
  "make": "Apollo",
  "model": "Evo",
  "year": 2017,
  "color": "Metalic Orange",
  "price": 5786,
  "isElectric": false,
  "createdAt": "2024-08-13T23:00:52.651Z",
  "updatedAt": "2024-08-14T03:36:19.639Z"
}
```

PROCEDIMIENTO - deleteOneCar

DAO – CONTROLLER: Método para eliminar un carro específico basado en su 'car_id' y controlador para eliminar un carro específico basado en su 'car_id', en caso de que no regresa un mensaje de error.

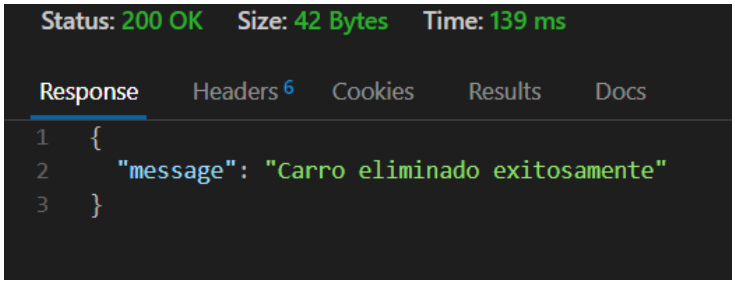
```
carsDao.findOneDelete = async (id) => {
    return await Cars.findOneAndDelete({ car_id: id });
};
```

```
export const deleteOneCar = (req, res) => {
    carsDao.findOneDelete(req.params.id)
        .then(car => {
            if (car)
                res.json({ message: "Carro eliminado exitosamente" });
            else
                res.json({ message: "Carro no encontrado para eliminar" });
        })
        .catch(err => {
            res.json({ message: err });
        });
};
```

ROUTER - Ruta para eliminar un carro específico por su 'car_id'.

```
router.delete("/deleteOneCar/:id", deleteOneCar);
```

RESPUESTA:



Status: 200 OK Size: 42 Bytes Time: 139 ms

Response Headers 6 Cookies Results Docs

```
{
  "message": "Carro eliminado exitosamente"
}
```