**db.config.js:**

module.exports = {

HOST: "localhost",

USER: "root",

PASSWORD: "DS433",

DB: "mydb"

};

**db.js:**

const mysql = require("mysql");

const dbConfig = require("../config/db.config.js");

// Create a connection to the database

const connection = mysql.createConnection({

host: dbConfig.HOST,

user: dbConfig.USER,

password: dbConfig.PASSWORD,

database: dbConfig.DB

});

// open the MySQL connection

connection.connect(error => {

if (error) throw error;

console.log("Successfully connected to the database.");

});

module.exports = connection;

**book.model.js:**

const sql = require("./db.js");

// constructor

const Book = function (book) {

this.title = book.title;

this.description = book.description;

this.published = book.published;

};

//creating book

Book.create = (newBook, result) => {

sql.query("INSERT INTO books SET ?", newBook, (err, res) => {

if (err) {

console.log("error: ", err);

result(err, null);

return;

}

console.log("created book: ", { id: res.insertId, ...newBook });

result(null, { id: res.insertId, ...newBook });

});

};

//finding book by id

Book.findById = (id, result) => {

sql.query(`SELECT \* FROM books WHERE id = ${id}`, (err, res) => {

if (err) {

console.log("error: ", err);

result(err, null);

return;

}

if (res.length) {

console.log("found book: ", res[0]);

result(null, res[0]);

return;

}

// not found Book with the id

result({ kind: "not\_found" }, null);

});

};

//get all books

Book.getAll = (title, result) => {

let query = "SELECT \* FROM books";

if (title) {

query += ` WHERE title LIKE '%${title}%'`;

}

sql.query(query, (err, res) => {

if (err) {

console.log("error: ", err);

result(null, err);

return;

}

console.log("books: ", res);

result(null, res);

});

};

//get all the published book

Book.getAllPublished = result => {

sql.query("SELECT \* FROM books WHERE published=true", (err, res) => {

if (err) {

console.log("error: ", err);

result(null, err);

return;

}

console.log("books: ", res);

result(null, res);

});

};

//update the existing book

Book.updateById = (id, book, result) => {

sql.query(

"UPDATE books SET title = ?, description = ?, published = ? WHERE id = ?",

[book.title, book.description, book.published, id],

(err, res) => {

if (err) {

console.log("error: ", err);

result(null, err);

return;

}

if (res.affectedRows == 0) {

// not found book with the id

result({ kind: "not\_found" }, null);

return;

}

console.log("updated book: ", { id: id, ...book });

result(null, { id: id, ...book });

}

);

};

//Remove book of the given id

Book.remove = (id, result) => {

sql.query("DELETE FROM books WHERE id = ?", id, (err, res) => {

if (err) {

console.log("error: ", err);

result(null, err);

return;

}

if (res.affectedRows == 0) {

// not found book with the id

result({ kind: "not\_found" }, null);

return;

}

console.log("deleted book with id: ", id);

result(null, res);

});

};

//Remove all the books

Book.removeAll = result => {

sql.query("DELETE FROM books", (err, res) => {

if (err) {

console.log("error: ", err);

result(null, err);

return;

}

console.log(`deleted ${res.affectedRows} books`);

result(null, res);

});

};

module.exports = Book;

**book.controller.js:**

const Book = require("../models/book.model.js");

// Create and Save a new Tutorial

exports.create = (req, res) => {

if (!req.body) {

res.status(400).send({

message: "Content can not be empty!"

});

}

// Create a Tutorial

const book = new Book({

title: req.body.title,

description: req.body.description,

published: req.body.published || false

});

// Save Tutorial in the database

Book.create(book, (err, data) => {

if (err)

res.status(500).send({

message:

err.message || "Some error occurred while creating the Tutorial."

});

else res.send(data);

});

};

// Retrieve all Books from the database (with condition).

exports.findAll = (req, res) => {

const title = req.query.title;

Book.getAll(title, (err, data) => {

if (err)

res.status(500).send({

message:

err.message || "Some error occurred while retrieving books."

});

else res.send(data);

});

};

// Find a single Book with a id

exports.findOne = (req, res) => {

Book.findById(req.params.id, (err, data) => {

if (err) {

if (err.kind === "not\_found") {

res.status(404).send({

message: `Not found Book with id ${req.params.id}.`

});

} else {

res.status(500).send({

message: "Error retrieving Book with id " + req.params.id

});

}

} else res.send(data);

});

};

// Update a Book identified by the id in the request

exports.update = (req, res) => {

const bookId = req.params.id;

const updatedBookData = req.body;

if (!updatedBookData || Object.keys(updatedBookData).length === 0) {

return res.status(400).json({ message: "Content can not be empty!" });

}

Book.updateById(bookId, updatedBookData, (err, data) => {

if (err) {

if (err.kind === "not\_found") {

return res.status(404).json({ message: `Not found Book with id ${bookId}.` });

} else {

return res.status(500).json({ message: `Error updating Book with id ${bookId}.` });

}

} else {

return res.json({ message: "Book was updated successfully!" });

}

});

};

// Delete a Book with the specified id in the request

exports.delete = (req, res) => {

Book.remove(req.params.id, (err, data) => {

if (err) {

if (err.kind === "not\_found") {

res.status(404).send({

message: `Not found Book with id ${req.params.id}.`

});

} else {

res.status(500).send({

message: "Could not delete Tutorial with id " + req.params.id

});

}

} else res.send({ message: `Book was deleted successfully!` });

});

};

// Delete all Books from the database

exports.deleteAll = (req, res) => {

Book.removeAll((err, data) => {

if (err) {

res.status(500).send({

message:

err.message || "Some error occurred while deleting all books."

});

} else res.send({ message: "All books were deleted successfully!" });

});

};

**book.routes.js:**

module.exports = app => {

const books = require("../controllers/book.controllers.js");

var router = require("express").Router();

// Create a new Tutorial

router.post("/", books.create);

// Retrieve all Tutorials

router.get("/", books.findAll);

// Retrieve all published Tutorials

router.get("/published", books.findAllPublished);

// Retrieve a single Tutorial with id

router.get("/:id", books.findOne)

// Update a Tutorial with id

router.put("/:id", books.update);

// Delete a Tutorial with id

router.delete("/:id", books.delete);

// Delete all Tutorials

router.delete("/", books.deleteAll);

app.use('/api/books', router);

};

**app.js:**

const express = require("express");

const cors = require("cors");

const app = express();

var corsOptions = {

origin: "http://localhost:4200"

};

app.use(cors(corsOptions));

// parse requests of content-type - application/json

app.use(express.json());

// parse requests of content-type - application/x-www-form-urlencoded

app.use(express.urlencoded({ extended: true }));

// simple route

app.get("/", (req, res) => {

res.json({ message: "Welcome to book application." });

});

//require("./app/routes/tutorial.routes.js")(app);

// set port, listen for requests

const PORT = process.env.PORT || 8080;

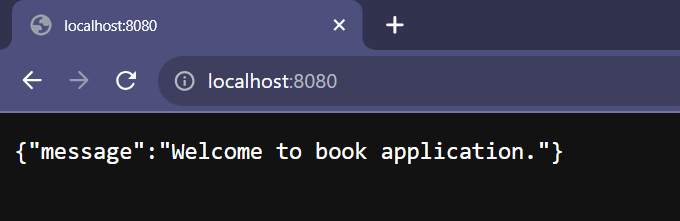
require("./routes/book.routes.js")(app)

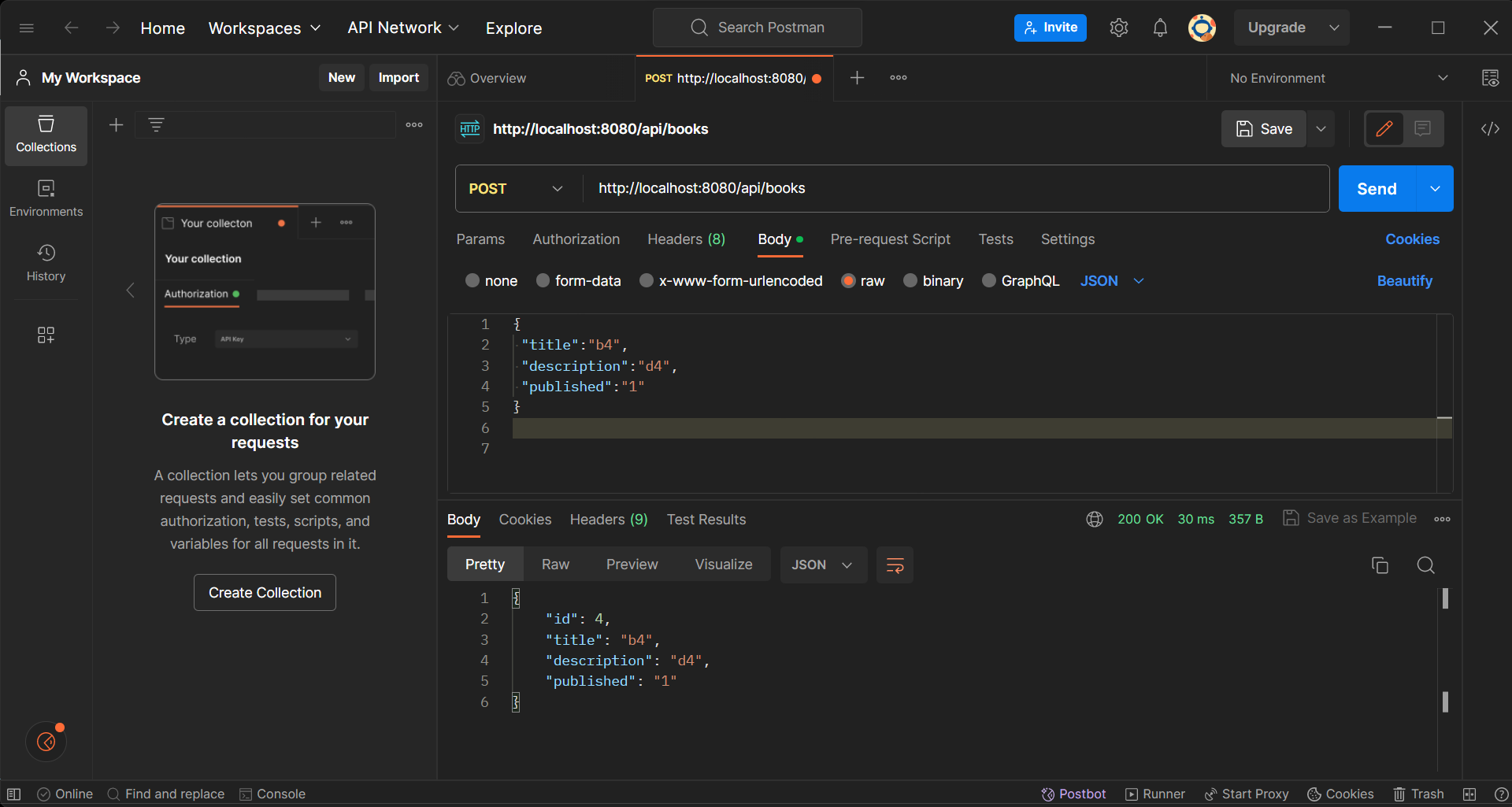
app.listen(PORT, () => {

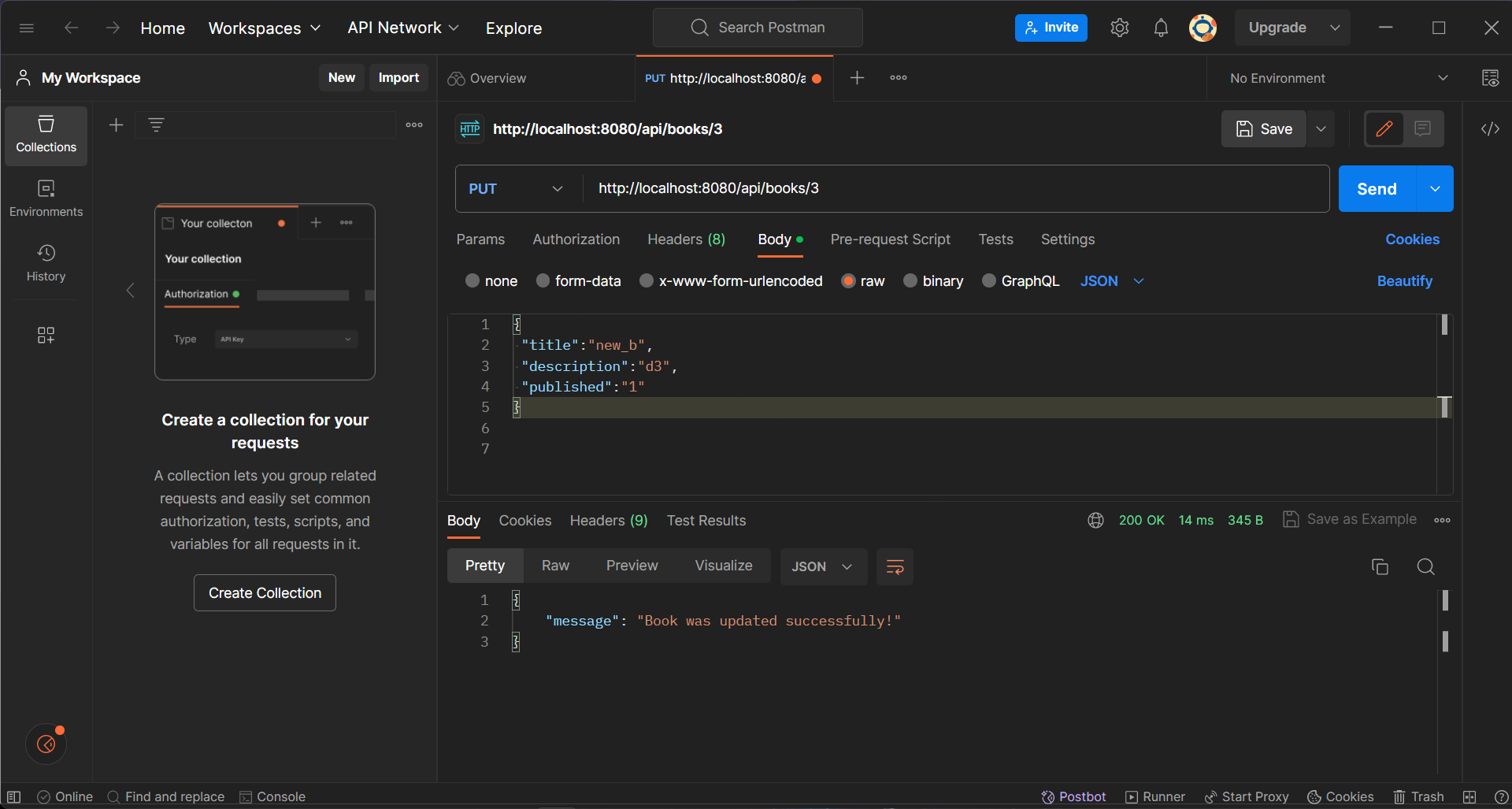
console.log(`Server is running on port ${PORT}.`);

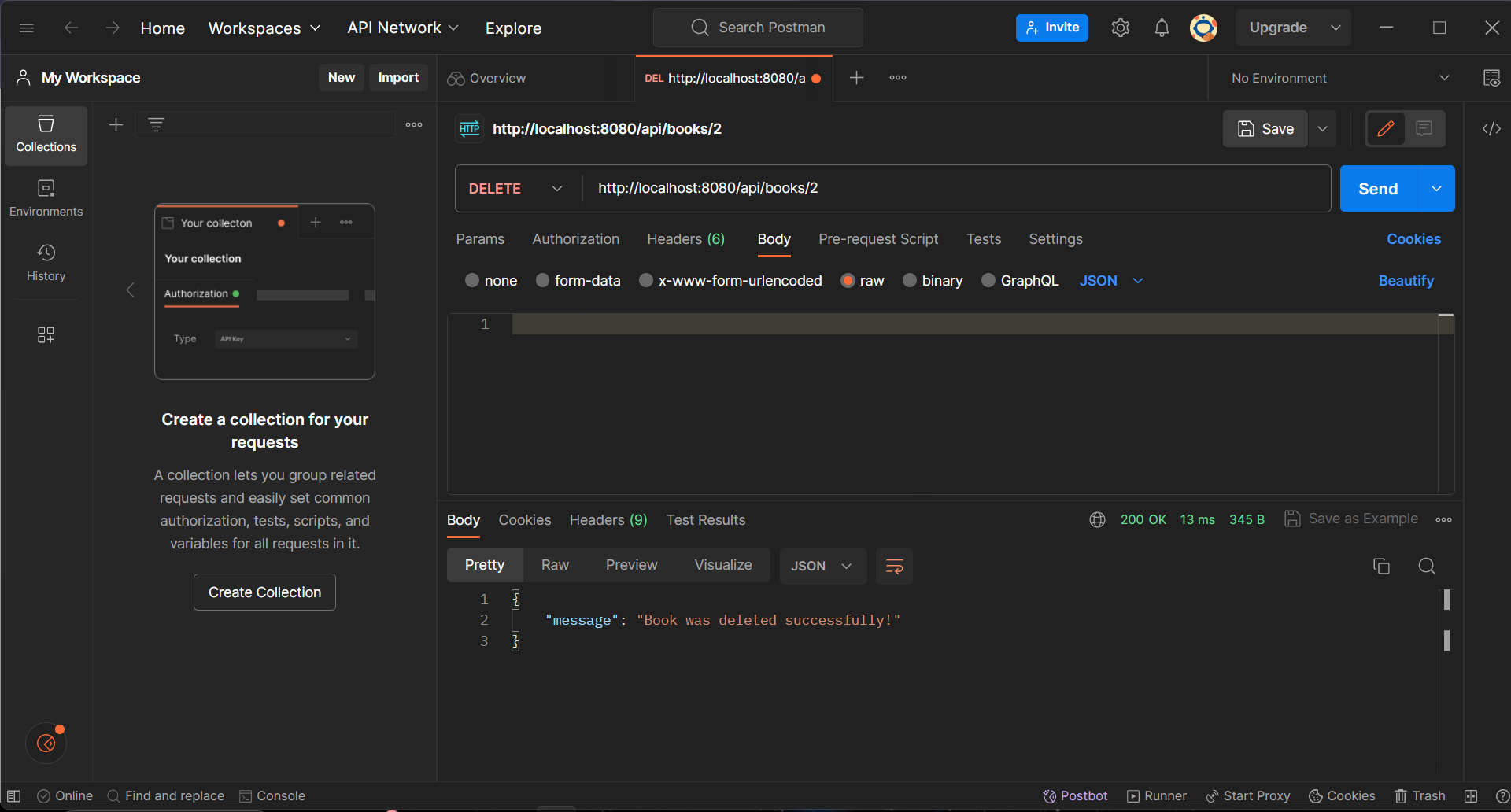
});

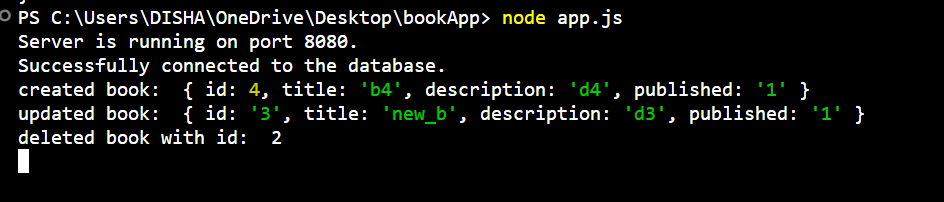
**OUTPUT:**

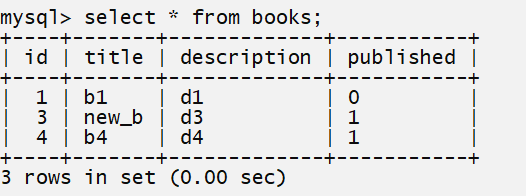
****

****

****

****

****

****