Backend-3

Agenda

- Middleware in Express
- DB Intution
- Mongo DB :
    1. MongoDB Atlas
    2. Concepts
    3. Mongoose
- env file

Middleware in Express

Middleware in Express.js are functions that execute during the lifecycle of a request to the server. Each middleware function can modify the request object (`req`), the response object (`res`), or end the request-response cycle. If it does not end the cycle, it must call the `next()` function to pass control to the next middleware function in the stack. Middleware functions can perform a range of tasks such as executing code, modifying the request and response objects, ending the request-response cycle, or calling the next middleware in the stack. It executed sequentially in the order they are defined.

req, res in middleware

In middleware function `res` is necessary without `res` no methods (GET,POST,PATCH & DELETE) will work.

Sequence

Middleware functions are executed in the order they are defined. This means that the sequence in which you apply middleware will determine how requests are processed.

Call next()

This function is used to pass control to the next middleware function in the stack. If a middleware function does not call `next()`, the request will hang, and subsequent middleware or route handlers will not be executed. Always ensure that `next()` is called unless the response is sent or the request is terminated.

Working like if condition

Middleware work like if condition where we need both methods and route.

Common Variable req & res

In middleware functions, `req` (request) and `res` (response) are common variables that are passed to each middleware function. These variables are crucial for interacting with the incoming HTTP request and sending a response back to the client.

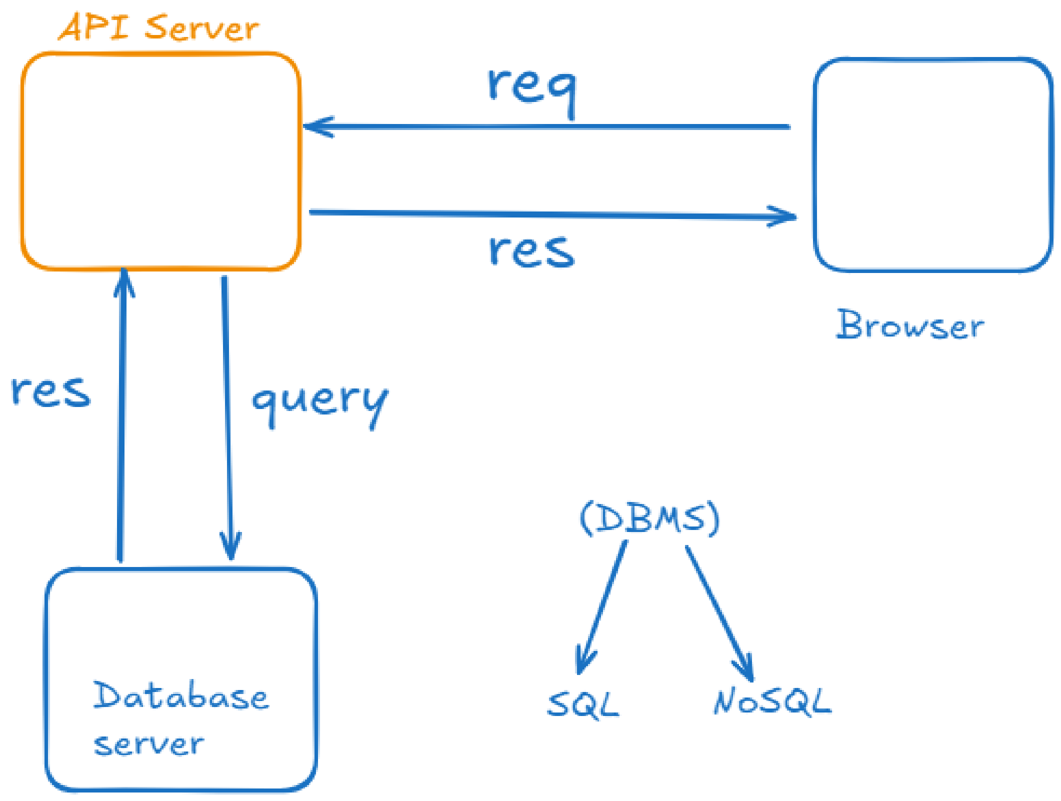
404 not found

`404 Not Found` middleware is used to handle requests that do not match any of the defined routes. This middleware should be placed after all other routes and middleware to catch any requests that fall through the cracks.

Database

Databases play a crucial role in storing, retrieving, and managing data for applications. Databases can be categorized into two main types: relational (SQL) and non-relational (NoSQL).

1. Relational Databases (SQL) : Relational databases store data in tables with rows and columns, using SQL (Structured Query Language) for querying and managing data.
2. Non-Relational Databases (NoSQL) : Non-relational databases store data in various formats like key-value pairs, documents, or graphs. They are designed for scalability and flexibility, making them suitable for applications with large volumes of unstructured or semi-structured data.



MongoDB

MongoDB is a popular NoSQL database that is widely due to its flexibility, scalability, and performance. It provides a document-oriented data model, where data is stored in flexible JSON-like documents.

MongoDB Atlas

- **Free Tier:** Offers a free tier with limited resources (e.g., 500MB storage, 100 read operations per second).
- **Sign Up:** Create a free account on the MongoDB Atlas website.
- **Create Cluster:** Follow the steps to create a new cluster.
- **Choose Free Tier:** Select the free tier plan.

Mongoose

It provides a convenient way to interact with MongoDB databases using a more object-oriented approach. Mongoose defines models, which represent the structure of your data, and provides methods for creating, reading, updating, and deleting documents.

Installation: Install Mongoose using npm:

```
npm install mongoose
```

db.js

```
const mongoose = require("mongoose");

const dotenv = require("dotenv")
dotenv.config({ path: "./Lecture_3_middleWare_db/.env" });

const dbLink = `mongodb+srv://${process.env.DB_USERNAME}
:${process.env.DB_PASSWORD}@cluster0.zc5df.mongodb.net/?
retryWrites=true&w=majority&appName=Cluster0`;

mongoose.connect(dbLink)
    .then(function (connection) {
        console.log("connected to db")
    }).catch(err => console.log(err))
```

Then it will connected to db.

Dotenv

dotenv is a popular package that allows us to manage environment variables in your application. It loads environment variables from a `.env` file into the process environment, making it easier to manage sensitive data and configuration settings.

Installation

```
npm install dotenv --save
```

.env

```
DB_USERNAME="your userid"
DB_PASSWORD="your password"
```