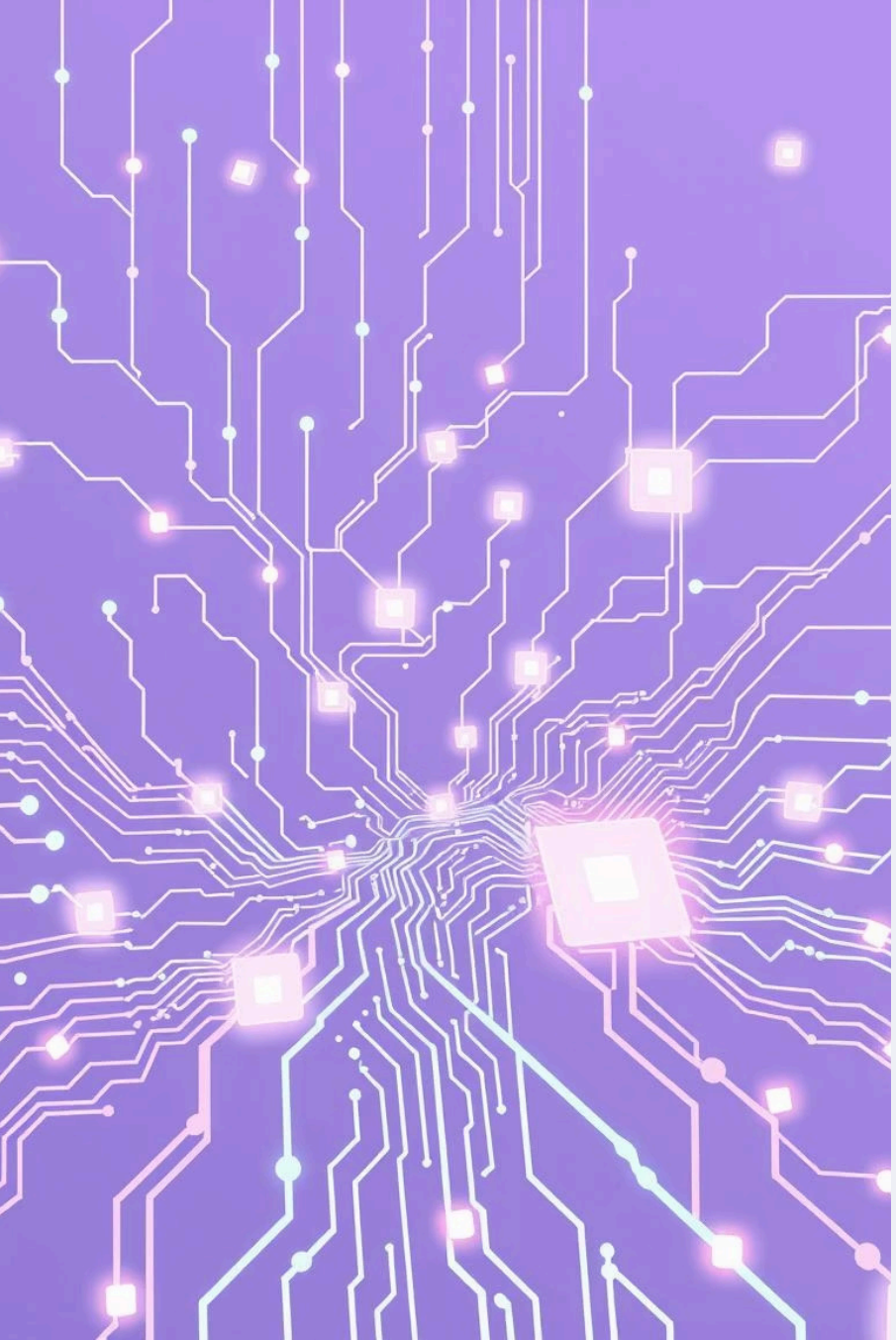


Mastering the MERN Stack: Day 2

Welcome to Day 2 of our MERN stack journey! Today, we dive deep into the heart of backend development: building robust and scalable RESTful APIs with Node.js, Express, and MongoDB.

Today's Learning Objectives

Understand REST Architecture

Grasp the core principles of RESTful design and its importance.

Build CRUD APIs with Express

Develop Create, Read, Update, Delete functionalities for your applications.

Connect Node.js with MongoDB

Utilize Mongoose for seamless database integration.

Design Basic Data Models

Learn to structure your data effectively with schemas.

Test APIs with Postman

Master the essential tool for validating your API endpoints.

Introduction to RESTful APIs

What is an API?

- A software intermediary that allows two applications to talk to each other.
- Acts as a bridge between client (e.g., a web browser) and server.
- Enables seamless communication between different software systems.

Documentation: https://restfulapi.net/

What is REST?

- **RE**presentational **S**tate **T**ransfer.
- An architectural style for designing networked applications.
- It's stateless, meaning each request from client to server contains all information needed.
- Leverages standard HTTP methods for operations.

> 📌 Analogy: Think of REST APIs like interacting with a waiter in a restaurant. You make a specific request (e.g., "GET me the menu"), and they bring you the desired result, without needing to remember your previous orders.

Key Concepts of REST

Resource

Any information the server can provide. Usually mapped to a database collection or table (e.g., **/users**, **/products**).

HTTP Methods

- **GET:** Retrieve data
- **POST:** Create new data

HTTP Methods (Cont.)

- **PUT/PATCH:** Update existing data
- **DELETE:** Remove data

While GraphQL offers an alternative, REST remains the dominant standard for most web services due to its simplicity and widespread adoption.

Building RESTful CRUD APIs

Mastering CRUD (Create, Read, Update, Delete) operations is fundamental to API development. Let's outline typical endpoints for a **User** resource:

HTTP Method + Path	Description
GET /api/users	Retrieve a list of all users.
GET /api/users/:id	Retrieve a specific user by their unique ID.
POST /api/users	Create a new user entry in the database.
PUT /api/users/:id	Update an existing user's data by their ID.
DELETE /api/users/:id	Remove a user entry from the database by their ID.

These standardized patterns make your API predictable and easy to consume for client applications.

Knowledge Check

When should we use PUT vs PATCH requests?

Connecting to MongoDB with Mongoose

What is Mongoose?

Mongoose is an **Object Data Modeling/Mapper (ODM)** library for Node.js. It provides a straightforward, schema-based solution to model your application data for MongoDB.

- Simplifies interactions with MongoDB.
- Handles schema validation, ensuring data consistency.
- Provides powerful query building capabilities.
- Manages relationships between different data models.

Think of Mongoose as the translator between your Node.js application and your MongoDB database, making data management much more intuitive.

Basic Data Modeling

Before storing data, you need to define its structure. Mongoose schemas are crucial for this.

Define Schemas

Use Mongoose schemas to enforce structure and validation rules for your documents (e.g., user schema with name, email, password fields).

```
const userSchema = new
mongoose.Schema({
  name: String,
  email: { type: String, required:
true, unique: true },
  age: Number
});
```

Create Models

Models are compiled from schemas and act as constructors for documents. They provide an interface for database operations.

```
const User =
mongoose.model('User',
userSchema);
```

Enforce Consistency

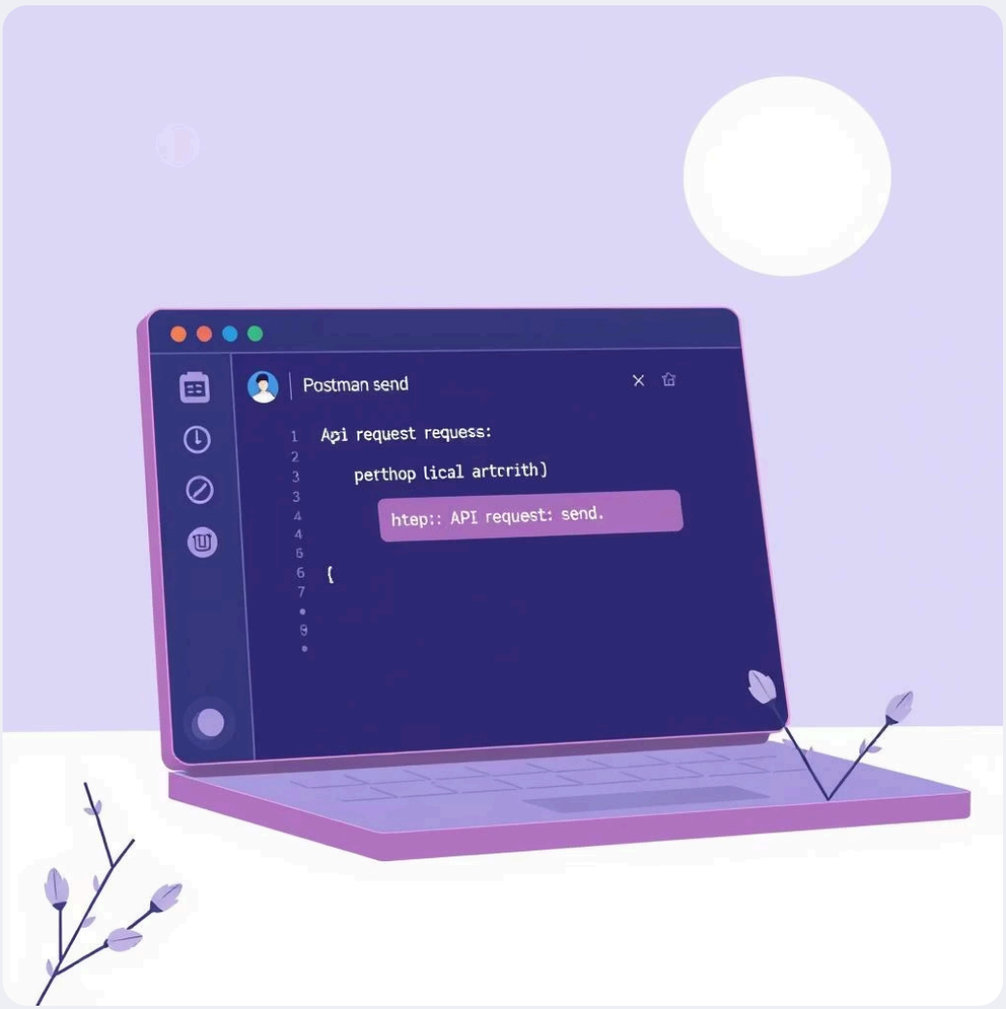
Schemas ensure that all documents adhere to a predefined structure, preventing common data integrity issues and making your application more robust.

Knowledge Check

When should we use Mongoose (or any other ODM/ORM)?

Testing APIs with Postman

Postman is an indispensable tool for developing and testing APIs. It provides a user-friendly graphical interface to send HTTP requests.



What is Postman?

- A powerful GUI client for making HTTP requests.
- Allows you to send requests (GET, POST, PUT, DELETE) to your API endpoints.
- Helps inspect responses, including status codes and data.

How to Test Your API:

1. Launch your Express server (e.g., npm run dev).
2. Open Postman and create a new request.
3. Send **POST** requests to **/api/users** with a JSON body to create new users.
4. Send **GET** requests to **/api/users** to retrieve data.
5. Experiment with **PUT** and **DELETE** requests using specific user IDs.