REST API

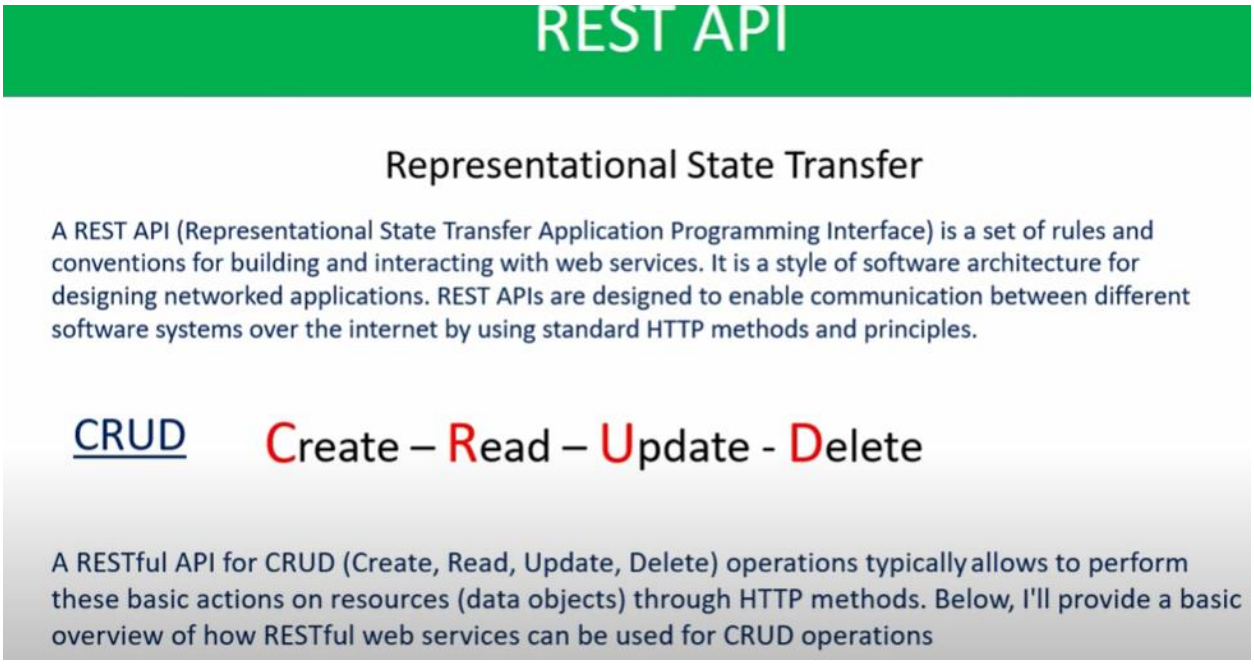
- What are API's
- What are REST API's
- What are Methods
- Routes
- MVC (Model View Controller)
- CRUD (Create Read Update Delete)

What is API

Application Programming Interface

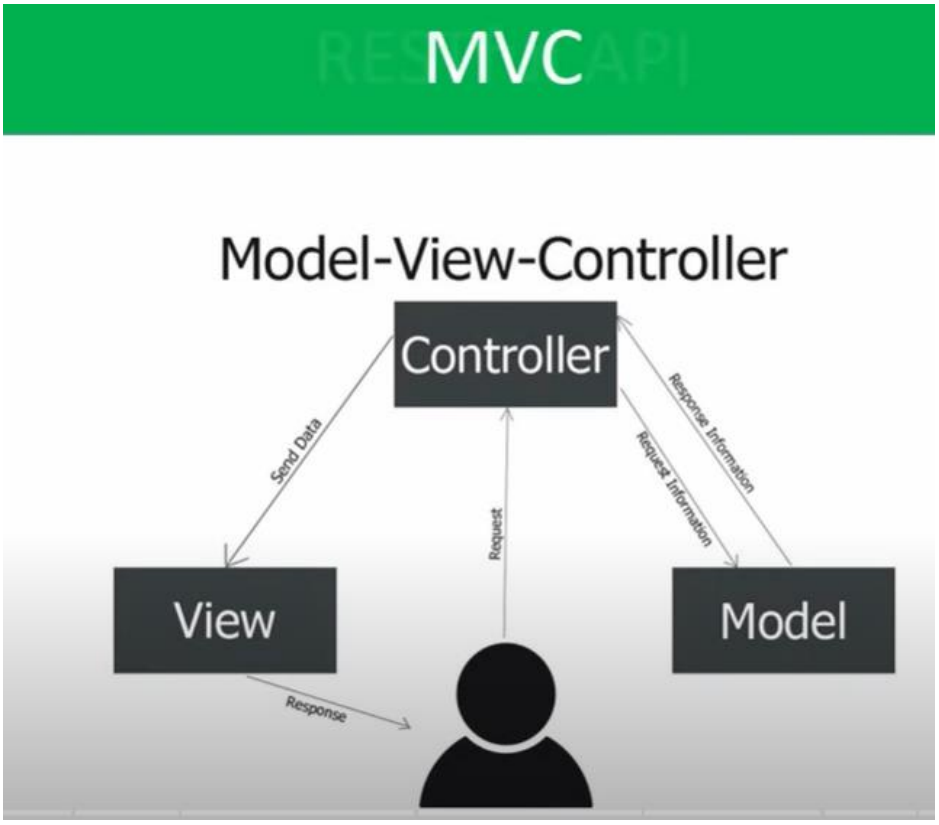
It is a set of rules and protocols that allows different software applications to communicate with each other.

TYPES OF API



WE WILL PERFORM CRUD OPERATIONS USING →

- GET - Get Data
- POST - Post or Store Data
- PUT - Update Data
- DELETE - Delete Data



Mongoose → is a library used for intraction mongodb

```
PS G:\MERN STACK\NodeMongodb> npm install mongoose
```

BodyParser → used to asses http request data

```
PS G:\MERN STACK\NodeMongodb> npm install body-parser
```

STEP 1 → CONNECT TO DATABASE(MONGODB)

```
const express = require("express");
const dotenv = require("dotenv");
const mongoose = require("mongoose");
const bodyParser = require("body-parser");

const app = express();

const PORT = process.env.PORT || 5000;

//to get ling from .env
dotenv.config();

//connection to mongodb
mongoose
  .connect(process.env.MONGO_URI)
  .then(() => {
    console.log("MongoDB Connected Successfully");
  })
  .catch((error) => {
    console.log(`${error}`);
  });

app.listen(PORT, () => {
  console.log(`Server Started and running at ${PORT}`);
});
```

The `then` and `catch` methods are part of the Promise API in JavaScript

The `then` method is used to handle the successful completion of a Promise. It takes up to two arguments: a callback function for the resolved case of the Promise, and optionally another callback function for handling any errors (although using `catch` is more common for error handling).

```
Server Started and running at 5000
MongoDB Connected Successfully
```

STEP 2 → CREATE models folder and create Employee.js in it

** we have create Schema

** in which file we are are going to create schema of database file name should start with capital and it should be singular bcoz in mongodb it will create plural collection for single

Employee Model (`Employee.js`):

- Defines the structure of the employee documents in the MongoDB database.
- Specifies required fields (`name` and `email`) and optional fields (`phone` and `city`).

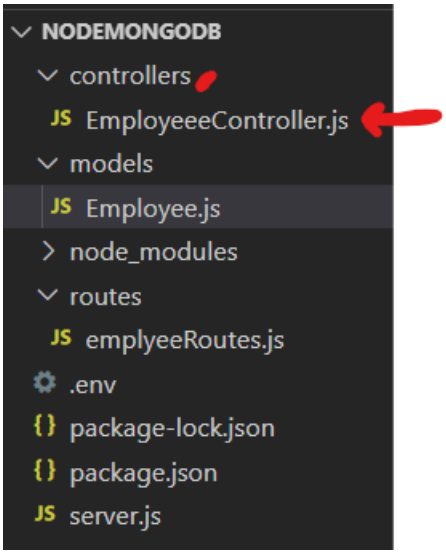
```
const mongoose = require("mongoose");

//creating schema
const employeeSchema = new mongoose.Schema({
  name: {
    type: String,
    require: true, // validation --> true means we have to enter complosory
  },
  email: {
    type: String,
    require: true,
  },
  phone: {
    type: Number,
    default: false, // no need complusory
  },
  city: {
    type: String,
  },
});

module.exports = mongoose.model("Employee", employeeSchema);
```

Step-3

Create controlers folder and create EmployeeeController.js



Employee Controller (`EmployeeController.js`):

- Implements the logic to create a new employee record.
- Validates and saves the employee data received from the client to the database.
- Sends appropriate HTTP responses based on the operation's success or failure.

llers > JS EmployeeeController.js > ...

```javascript
// importing from Employee.js
const Employee = require("../models/Employee");
//.. to change the folder

const createEmployee = async (req, res) => {
  try {
    const { name, email, phone, city } = req.body;

    const employee = new Employee({
      name,
      email,
      phone,
      city,
    });

    await employee.save();
    res.status(201).json(employee);
    //201 indicates success
  } catch (error) {
    console.log("there is an error : ", error);
    res.status(500).json({ message: "server error" });
  }
};

module.exports = { createEmployee };
```

Step – 4

Create router folder → in it → employeeRouters.js

This code defines the routes for handling employee-related HTTP requests using Express Router. It specifies the endpoint for adding a new employee and maps it to the appropriate controller function.

```javascript
const express = require("express");
const router = express.Router();
const employeeController = require("../controllers/EmployeeeController");
const Employee = require("../models/Employee");

router.post("/add-emp", employeeController.createEmployee);

module.exports = router;
```

```javascript
router.post("/add-emp", employeeController.createEmployee);
```

- Defines a POST route `/add-emp`.
- When a POST request is made to `/add-emp`, it calls the `createEmployee` function from the `employeeController`.

Finally adding this router to middle ware

The line `app.use("/employees", employeeRoutes);` is used to set up middleware in your Express application to handle routes related to employees. Here's a detailed breakdown of what this middleware does:

Using Middleware:

```javascript
app.use(bodyParser.json());
```

- Parses incoming request bodies in JSON format.

```javascript
const express = require("express");
const dotEnv = require("dotenv");
const mongoose = require("mongoose");
const bodyParser = require("body-parser");
const employeeRoutes = require("./routes/emplyeeRoutes"); //taking rotes

const app = express();

const PORT = process.env.PORT || 5000;

dotEnv.config();
// to convert to json
app.use(bodyParser.json());

mongoose
  .connect(process.env.MONGO_URI)
  .then(() => {
    console.log("MongoDB connected Successfully");
  })
  .catch((error) => {
    console.log(`ERROR : ${error}`);
  });

//middleware
//arrg-1->defining url
//arg-> where we are getting routes
app.use("/employees", employeeRoutes);

app.listen(PORT, () => {
  console.log(`Server Started Running at ${PORT}`);
});
```

Now we have add any thing to mango db like rows we have open post man → enter url → post request

** postman used to tesp API

POST localhost:5000/employees/add-emp Send

5000->port

Employees → middleware in server.js

add-emp → router → in employeeRouter.js

raw JSON

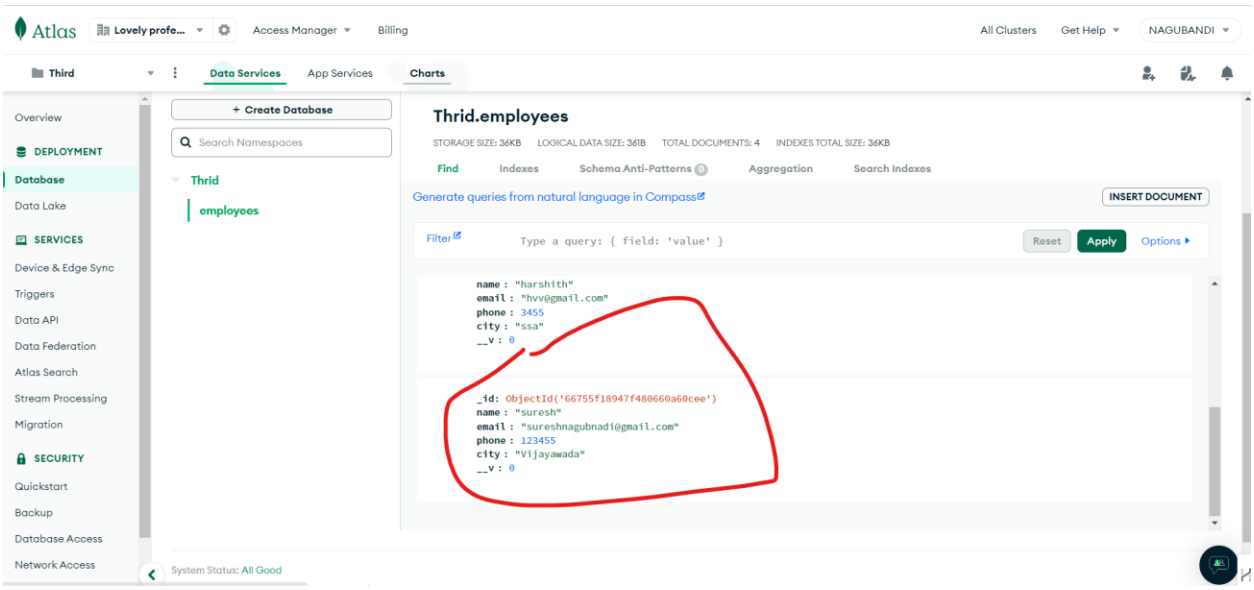
```
{
    "name":"suresh",
    "email":"sureshnagubnadi@gmail.com",
    "phone":123455,
    "city":"Vijayawada"
}
```

Body 201 Created 710 ms 37(

Pretty Raw Preview Visualize JSON

```
{
    "name": "suresh",
    "email": "sureshnagubnadi@gmail.com",
    "phone": 123455,
    "city": "Vijayawada",
    "_id": "66755f18947f480660a60cee",
    "__v": 0
}
```

Now see this in mongo db new record is added


Atlas
Lovely profe...
Access Manager
Billing
All Clusters
Get Help
NAGUBANDI
Third
Data Services
App Services
Charts
Overview
DEPLOYMENT
Database
Data Lake
SERVICES
Device & Edge Sync
Triggers
Data API
Data Federation
Atlas Search
Stream Processing
Migration
SECURITY
Quickstart
Backup
Database Access
Network Access
+ Create Database
Search Namespaces
Thrid
employees
Thrid.employees
STORAGE SIZE: 36KB
LOGICAL DATA SIZE: 361B
TOTAL DOCUMENTS: 4
INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
Generate queries from natural language in Compass
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
name : "harshith"
email : "hvv@gmail.com"
phone : 3455
city : "ssa"
__v : 0
_id: ObjectId('66755f18947f480660a60cee')
name : "suresh"
email : "sureshnagubnadi@gmail.com"
phone : 123455
city : "Vijayawada"
__v : 0
System Status: All Good