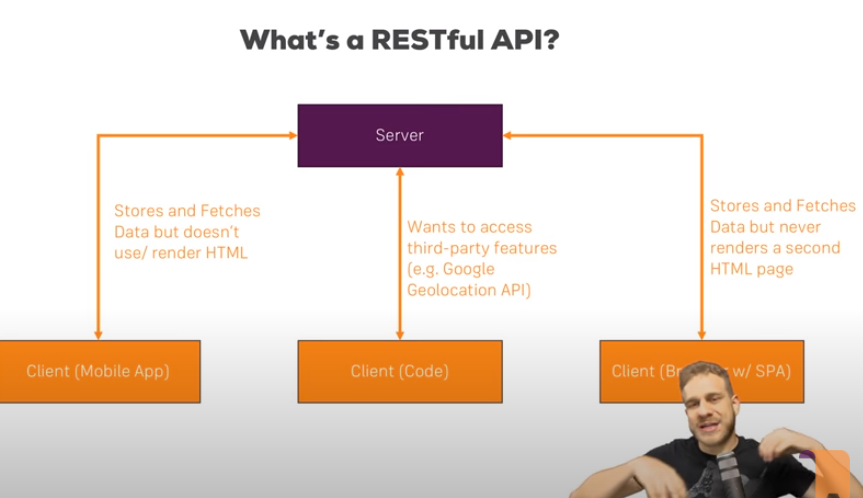
REST API

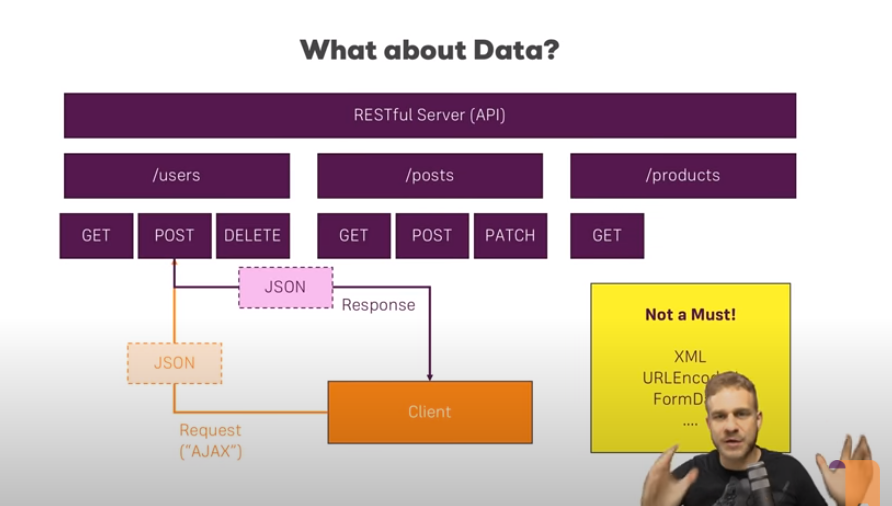
It stands for representational state transfer

Used to exchange data between client and server, like sending and fetching data to and from server. Data mostly exchange in JSON format, but not mandatory

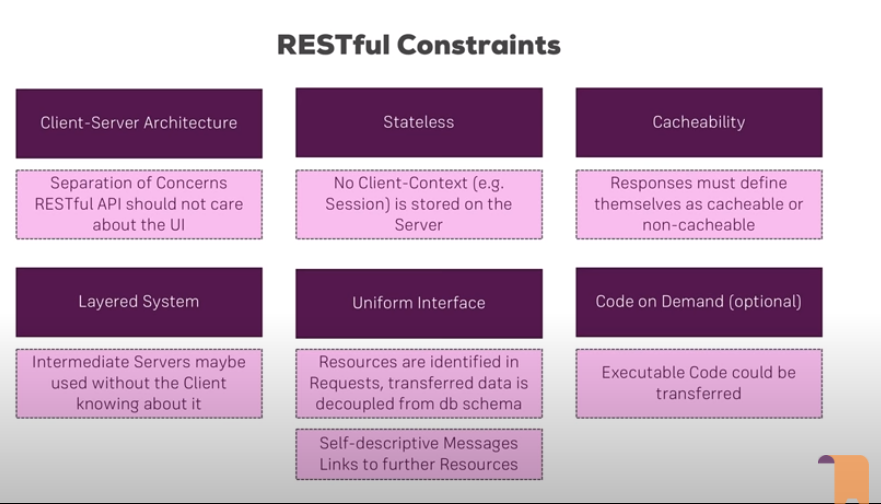
Does not render html pages

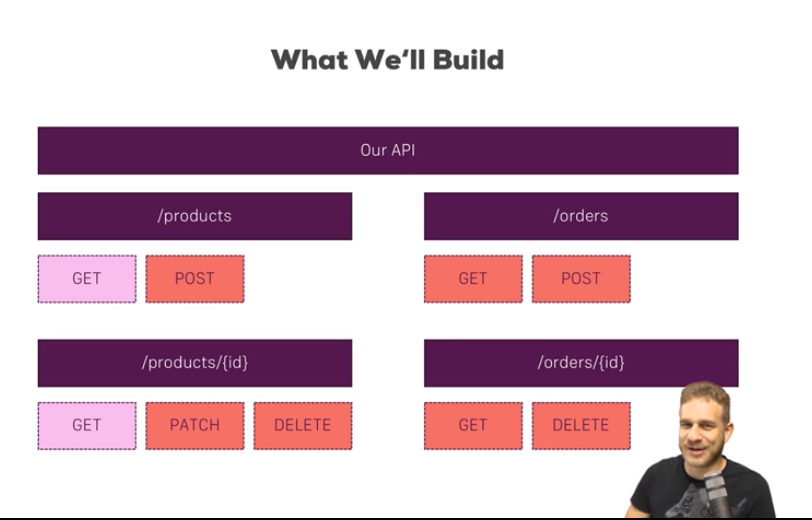


It is a stateless backend, does not care of individual client connected to it.



RESTful Constraints:





Dependencies to install:

Npm I –save-dev morgan (to show logs on terminal or console, not necessary)

Npm init

Npm I express

Npm I –save-dev nodemon

(to make app automatically save, and don’t need to restart it everytime)

After adding it, update package.json file

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "start": "nodemon server.js"

  },

And now app will be start like npm start

SIMPLE GET AND POST METHOD:

Server.js

const http = require('http');

const app = require('./app')

const port = process.env.PORT || 3000

const server = http.createServer(app);

server.listen(port);

app.js

const express = require('express')

const app = express();

const productRoutes = require('./api/routes/products')

//middleware

app.use('/products', productRoutes);

module.exports = app;

products.js

const express = require('express')

const router = express.Router(); //to create routes

router.get('/', (req, res, next)=>{

    res.status(200).json({

        message: "Handling GET requests to /products"

    })

})

router.post('/', (req, res, next)=>{

    const product = {

        name: req.body.name,

        price: req.body.price

    }

    res.status(201).json({

        message: "Handling POST requests to /products",

        createdProduct: product

    })

})

//for particular product in products

router.get('/:productId', (req, res, next)=>{

    const id = req.params.productId;

    if(id === 'special'){

        res.status(200).json({

            message: 'You discover the special ID',

            id: id

        })

    }

    else {

        res.status(200).json({

            message: 'You passed an ID'

        })

    }

})

router.patch('/:productId', (req, res, next)=>{

    res.status(200).json({

        message: 'Updated product!'

    })

})

router.delete('/:productId', (req, res, next)=>{

    res.status(200).json({

        message: 'Deleted product'

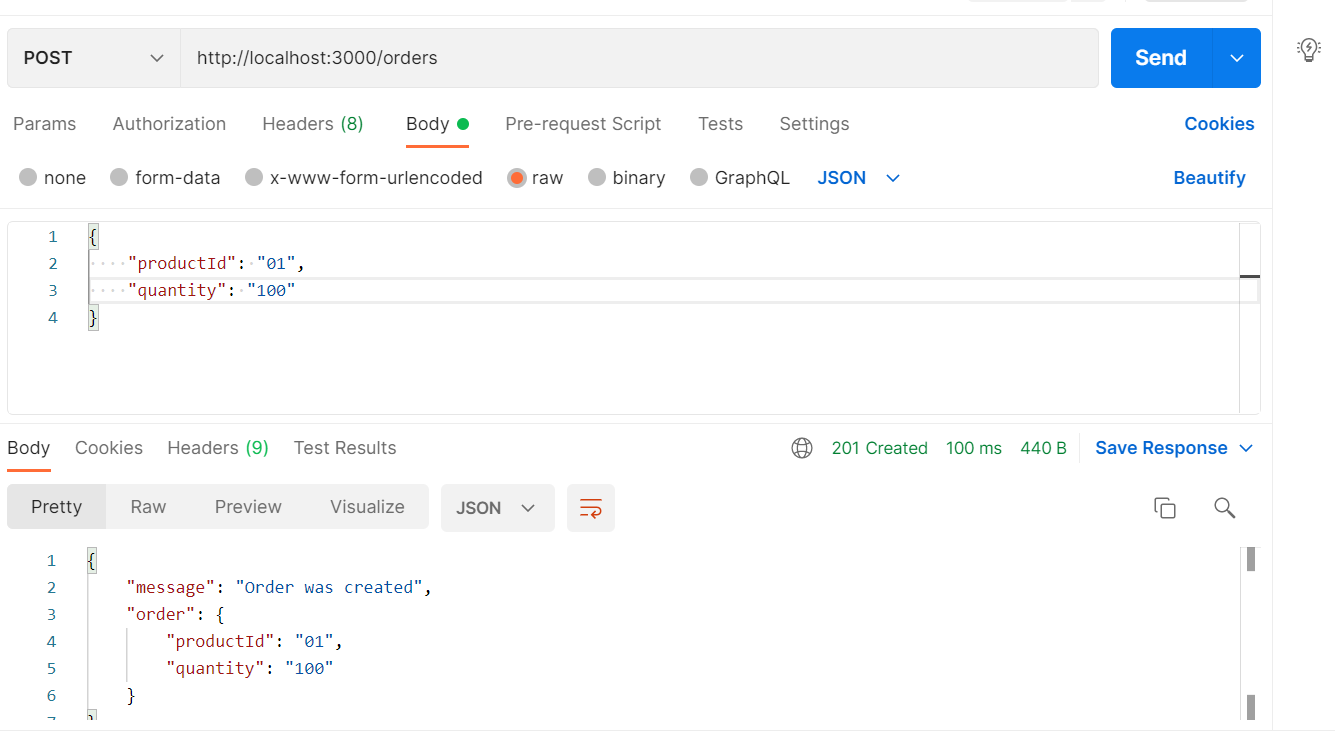
    })

})

module.exports = router;

Till here, we have covered get, post, delete and update route(patch). This is how it works

We can provide name and price in body section of postman and selecting raw and json



Now adding another parent route which is for the orders, creating file orders.js which contains all the routes and which is called in app.js in app.use

Order.js

const express = require('express')

const router = express.Router(); //to create routes

router.get('/', (req, res, next)=>{

    res.status(200).json({

        message: "Orders were fetched"

    })

})

router.post('/', (req, res, next)=>{

    const order = {

        productId: req.body.productId,

        quantity: req.body.quantity

    }

    res.status(201).json({

        message: "Order was created",

        order: order

    })

})

//for particular order in orders

router.get('/:orderId', (req, res, next)=>{

    res.status(200).json({

            message: 'Order details',

            orderId: req.params.orderId

        })

})

router.delete('/:orderId', (req, res, next)=>{

    res.status(200).json({

        message: 'Order deleted',

        orderId: req.params.orderIds

    })

})

module.exports = router;

Same, we can pass json object in postman like we did for products

Calling in app.js

const express = require('express')

const app = express();

const productRoutes = require('./api/routes/products')

const orderRoutes = require('./api/routes/orders')

//middleware

app.use('/products', productRoutes);

app.use('/orders', orderRoutes);

module.exports = app;

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Till above, basic routing completed

UPDATED APP.js with error handling (Use this for now):

const express = require('express')

const morgan = require('morgan')

const app = express();

const productRoutes = require('./api/routes/products')

const orderRoutes = require('./api/routes/orders')

//morgan shows log in terminal or console

app.use(morgan('dev'))

//middleware

app.use('/products', productRoutes);

app.use('/orders', orderRoutes);

//middleware for error handling

//if error not caught by above routes, then below works

//This one is for 404 error, and shows error Not Found

app.use((req, res, next)=>{

    const error = new Error("Not Found");

    error.status = 404;

    next(error);

})

//This one is for all kind of errors, or 500 error

app.use((error, req, res, next)=>{

    res.status(error.status || 500);

    res.json({

        error:{

            message: error.message

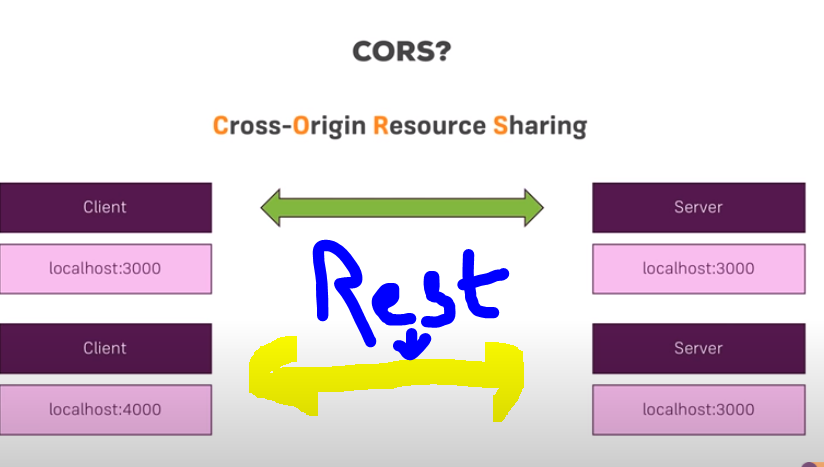
        }

    })

})

module.exports = app;

CORS:

In rest, we have client and server both on different url, so client says that why you are going to different server, to avoid that security issues, we provide headers to avoid those cors errors and asked client to access different server url as well

NOW UPDATED APP.JS with CORS:

Below is the app.js almost have everything, but with db

const express = require('express')

const morgan = require('morgan')

const app = express();

const productRoutes = require('./api/routes/products')

const orderRoutes = require('./api/routes/orders')

//middleware

//morgan shows log in terminal or console

app.use(morgan('dev'))

//body parser which is now replaced by express is used to parse body of your choice

app.use(express.urlencoded({extended: false}))//parse simple bodies of url encoded data

app.use(express.json());//this is to extract json data and make it readable

//Insurance to prevent cors error

app.use((req, res, next)=>{

    res.header("Access-Control-Allow-Origin", "\*");

    res.header("Access-Control-Allow-Headers", 'Origin, X-Requested-With, Content-Type, Accept, Authorization');

    if (req.method === 'OPTIONS'){

        res.header("Access-Control-Allow-Methods", "PUT, POST, PATCH, DELETE, GET");

        return res.status(200).json({});

    }

    next();

})

//middleware to handle routes

app.use('/products', productRoutes);

app.use('/orders', orderRoutes);

//middleware for error handling

//if error not caught by above routes, then below works

//This one is for 404 error, and shows error Not Found

app.use((req, res, next)=>{

    const error = new Error("Not Found");

    error.status = 404;

    next(error);

})

//This one is for all kind of errors, or 500 error

app.use((error, req, res, next)=>{

    res.status(error.status || 500);

    res.json({

        error:{

            message: error.message

        }

    })

})

module.exports = app;

USE OF ATLAS MONGODB:

First update app.js by adding connections

const express = require('express')

const morgan = require('morgan')

const mongoose = require('mongoose')

const app = express();

const productRoutes = require('./api/routes/products')

const orderRoutes = require('./api/routes/orders')

//mongodb connection

mongoose.connect('mongodb+srv://admin:Dj21mrMwyo89gAHr@cluster0.dmxhm.mongodb.net/myFirstDatabase?retryWrites=true&w=majority',

{

    useNewUrlParser:true,

    useUnifiedTopology:true

});

//middleware

//morgan shows log in terminal or console

app.use(morgan('dev'))

//body parser which is now replaced by express is used to parse body of your choice

app.use(express.urlencoded({extended: false}))//parse simple bodies of url encoded data

app.use(express.json());//this is to extract json data and make it readable

//Insurance to prevent cors error

app.use((req, res, next)=>{

    res.header("Access-Control-Allow-Origin", "\*");

    res.header("Access-Control-Allow-Headers", 'Origin, X-Requested-With, Content-Type, Accept, Authorization');

    if (req.method === 'OPTIONS'){

        res.header("Access-Control-Allow-Methods", "PUT, POST, PATCH, DELETE, GET");

        return res.status(200).json({});

    }

    next();

})

//middleware to handle routes

app.use('/products', productRoutes);

app.use('/orders', orderRoutes);

//middleware for error handling

//if error not caught by above routes, then below works

//This one is for 404 error, and shows error Not Found

app.use((req, res, next)=>{

    const error = new Error("Not Found");

    error.status = 404;

    next(error);

})

//This one is for all kind of errors, or 500 error

app.use((error, req, res, next)=>{

    res.status(error.status || 500);

    res.json({

        error:{

            message: error.message

        }

    })

})

module.exports = app;

Now to store and play with db, first create schema for products.js with name product.js

const mongoose = require('mongoose');

const productSchema = mongoose.Schema({

    \_id : mongoose.Schema.Types.ObjectId,

    name: String,

    price: Number

})

module.exports = mongoose.model('Product', productSchema);

Now calling this schema in api products.js and update all apis with this Product

const express = require('express')

const router = express.Router(); //to create routes

const mongoose = require('mongoose')

const Product = require('../Schema/product');

router.get('/', (req, res, next)=>{

    Product.find()

           .exec()

           .then(docs =>{

               console.log(docs);

            //    if (docs.length >= 0){

                res.status(200).json(docs);

        //        }else{

        //            res.status(404).json({

        //                message: "No entries found"

        //            })

        //        }

           })

           .catch(err =>{

               console.log(err);

               res.status(500).json({

                   error:err

               })

           })

})

router.post('/', (req, res, next)=>{

    // const product = {

    //     name: req.body.name,

    //     price: req.body.price

    // }

    const product = new Product({

        \_id: new mongoose.Types.ObjectId(),

        name: req.body.name,

        price: req.body.price

    });

    product.save()

           .then(result=>{

                console.log(result);

                res.status(201).json({

                    message: "Handling POST requests to /products",

                    createdProduct: result

                })

            })

            .catch(err=> {

                console.log(err)

                res.status(500).json({

                    error: err

                })

            });

})

//for particular product in products

router.get('/:productId', (req, res, next)=>{

    const id = req.params.productId;

    Product.findById(id)

           .exec()

           .then(doc=>{

               console.log("From dataabase" + doc)

               if(doc){

                res.status(200).json(doc)

               } else{

                   res.status(404).json({message: 'No valid entry found for ID'})

               }

           })

           .catch(err=>{

               console.log(err);

               res.status(500).json({error: err})

            })

})

router.patch('/:productId', (req, res, next)=>{

    const id = req.params.productId;

    const updateOps = {};

    for (const ops of req.body){

        updateOps[ops.propName] = ops.value; //{name: req.body.newName, price: req.body.newPrice}

    }

    Product.updateOne({\_id:id}, { $set: updateOps})

            .exec()

            .then(result => {

                console.log(result);

                res.status(200).json(result)

            })

            .catch(err => {

                console.log(err);

                res.status(500).json({

                    error: err

                })

            })

    res.status(200).json({

        message: 'Updated product!'

    })

})

router.delete('/:productId', (req, res, next)=>{

    const id = req.params.productId

    Product.remove({\_id: id})

            .exec()

            .then(result => {

                res.status(200).json(result);

            })

            .catch(err => {

                res.status(500).json({

                    error: err

                })

            });

})

module.exports = router;