Celebrating Glorious Platinum Jubilee Year 2022

WALCHAND COLLEGE OF ENGINEERING

WLUG
COMMUNITY | KNOWLEDGE | SHARE

WALCHAND LINUX USERS' GROUP

TECHNOTWEET

UNLEASH THE FULL POTENTIAL

A CHANCE TO WIN
EXCITING PRIZES
LIMITED SEATS ONLY

29/04 **MEGA-EVENT** **30/04**

FRONT-END

- React Basics & Components
- React hooks
- API Integration & React Axios

BACK-END

- NodeJS basics & modules
- ExpressJS & Middleware
- MongoDB, GraphQL & REST API's

REGISTRATION ₹ 199 FEE

VENUE
MINI & MAIN CCF
9:30 AM

wcewlug.org

For any query
7020448836
8421006401

Miss D. A. Kolapkar
President
Walchand Linux Users' Group

Dr. M. A. Shah
HoD Computer Science Engineering

Dr. A. J. Umbarkar
HoD Information Technology and Staff Advisor

Dr. A. R. Surve
Chairperson ECAC and Staff Advisor

Dr. U.A. Dabade
I/C Director
Walchand College of Engineering

TECHNOTWEET

WLUG
COMMUNITY | KNOWLEDGE | SHARE

NODE EXPRESS

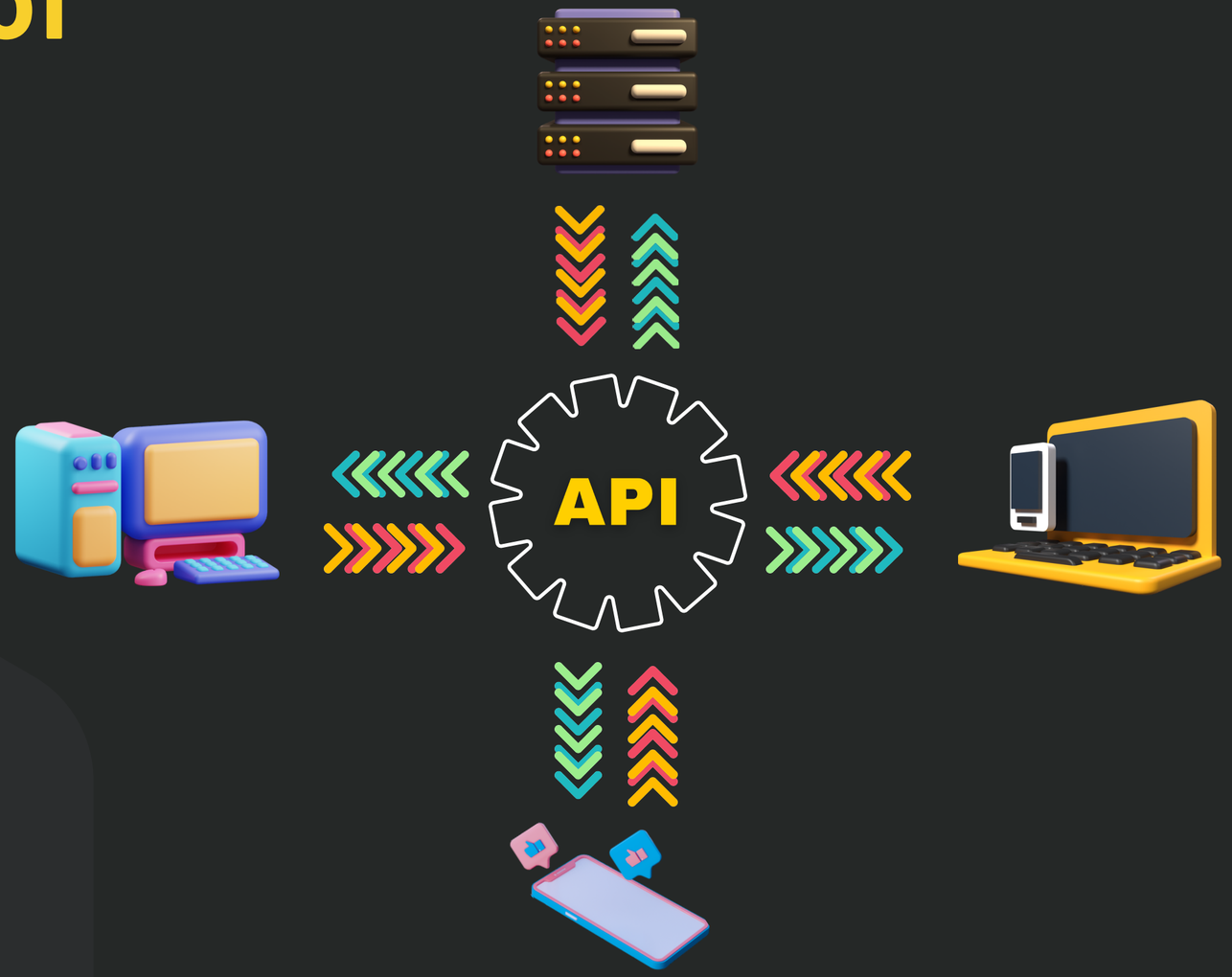
WLUG
COMMUNITY | KNOWLEDGE | SHARE

What is Backend ?

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Basic Components of Backend

- Server
- Database
- API
- Frameworks
- Authentication and Authorization
- Security



WLUG
COMMUNITY | KNOWLEDGE | SHARE

Node JS

- Is an open source cross-platform runtime environment for executing JS code

Features of Node JS

- Asynchronous and Event Driven
- Very Fast
- Single Threaded but Highly Scalable
- No buffering

Callback

- Callback is a function passed as argument to another function
- Callback function waits for result of previous function call and then execute another function call

```
function bill() {
    console.log(`Here is your bill !!`)
}

// function is passed as argument
function order(name , myCallback){
    console.log(`${name} for you`)
    myCallback()    // calling function
}

// calling function
order('coffee' , bill)
```

TECHNOTWEET

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Promises

- A Promise is a special JavaScript object
- It produces a value after an async operation completes successfully
- It produces an error if it doesn't complete successfully

```
// Creating promise object
let order = new Promise((resolve, reject) => {
    // promise fullfilled
    resolve("Coffee for you")

    // promise not fullfilled
    reject(new Error("Sorry coffee is not available"))
})

// Log the result
console.log(order)
```

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Async Await

- Async/Await is a cleaner syntax for working with Promises

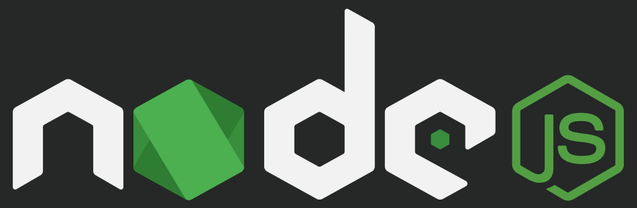
```
function order() {
    return new Promise( (resolve) => {
        setTimeout(() => {
            resolve('Coffee for you');
        }, 2000);
    });
}

// creating async function
async function waiter() {
    console.log('Order has been placed...');

    // wait for completion
    const res = await order();
    console.log(res);

    console.log('Here is your bill !!');
}

waiter();
```

Modules

- Blocks of encapsulated code
- Single file or a collection of multiples files/folders
- Re-usability

Types of Modules

- Core Modules
- Local Modules
- Third-party Modules



WLUG
COMMUNITY | KNOWLEDGE | SHARE

Os Module

- Provides a way to get information about the operating system

- System information like
  - User Info
  - Uptime
  - Os type
  - Free Memory

```
const os = require('os');

// info about current user
const user = os.userInfo()
console.log(user)

// method returns the system uptime in seconds
console.log(`The system uptime is ${os.uptime()} seconds`)

const currentOs = {
    name: os.type(),
    release: os.release(),
    totalMem: os.totalmem(),
    freeMem: os.freemem(),
}

console.log(currentOs)
```

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Path Module

- Provides utilities for working with file paths
- With methods like
  - path.sep
  - path.join()
  - path.basename()
  - path.resolve()

```
const path = require('path');

// platform-specific path segment separator.
console.log(path.sep);

// Join directory and file path
const filePath = path.join('/wlug', 'techno', 'tweet.txt')
console.log(filePath);

const base = path.basename(filePath)
console.log(base);

// To get the absolute path of a file
const absolute = path.resolve(__dirname, 'wlug', 'techno', 'tweet.txt')
console.log(absolute);
```

FS Sync

- Provides an API for working with the file system
- In Sync method task at hand is completed till completion
- New task starts after full execution of previous task

```
const fs = require('fs')

// Reading
const first = fs.readFileSync('./content/one.txt', 'utf8')
const second = fs.readFileSync('./content/two.txt', 'utf8')

// Writing
fs.writeFileSync(
    './content/res.txt', `Res = ${first} ${second}`
)

const res = fs.readFileSync('./content/res.txt', 'utf8')
// Printing
console.log(res)
```

WLUG
COMMUNITY | KNOWLEDGE | SHARE

FS Async

- In Async method you can start new task simultanously
- No need to wait for the completion of previous task

```
const fs = require('fs')

// Reading from file
fs.readFile('./content/res.txt' , 'utf-8' , (err , data) => {
    if (err) {
        console.log(err)
    }
    console.log(data)
})

// Appending to file
fs.appendFile('./content/res.txt' , ' Adding data' , (err, data) => {
    if (err) {
        console.log(err)
    }
    console.log("Data added")
})
```

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Fs Promises

- Fs promises = FS + Promises
- Makes code easier to read
- More structured way for async operations and error handling

```
const fs = require('fs').promises;

// Write data to a file
fs.writeFile('tweet.txt', 'Hi there, writing some data')
  .then(() => console.log('Data written successfully'))
  .catch((err) => console.error(err));

// Read data from a file
fs.readFile('tweet.txt', 'utf8')
  .then((data) => console.log(data))
  .catch((err) => console.error(err));
```

HTTP Module

- The HTTP module is a core module of Node.
- Designed to support many features of the HTTP protocol

```
// import node.js built-in http module
var http = require('http');
// creating server
http.createServer((req, res) => {
    // Write response to client
    res.write('Hello World!');
    // End the response
    res.end();
}).listen(4200); // Server object listen on port 4200
```

Node Package Manager

- Default package manager for Node
- World's largest Software Registry
- Is open source

NPM Basic Commands

```
npm init <flags>
#initializes npm for the project
#flag -y is used for using default values for prompts

npm i <package_name>@<version> <flags>

npm install axios
npm install axios@3.0.0
npm install githubname/reponame
npm install nodemon --save-dev
npm install uuid --save-optional
npm install date-fns --save-exact

npm uninstall axios
npm uninstall express --no-save

# instead of uninstall you can use remove, rm, r, un, unlink
```

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Package.json

```
{
  "name": "npm-modules",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "date-fns": "^2.29.3",
    "uuid": "^9.0.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.20"
  }
}
```

Node Modules

Third-party modules and their dependencies are kept under a common directory called Node Modules

NodeJS Events

- Node.js has a built-in module, called "Events"
- All event properties and methods are an instance of an EventEmitter object
- Create-, fire-, and listen for- your own events

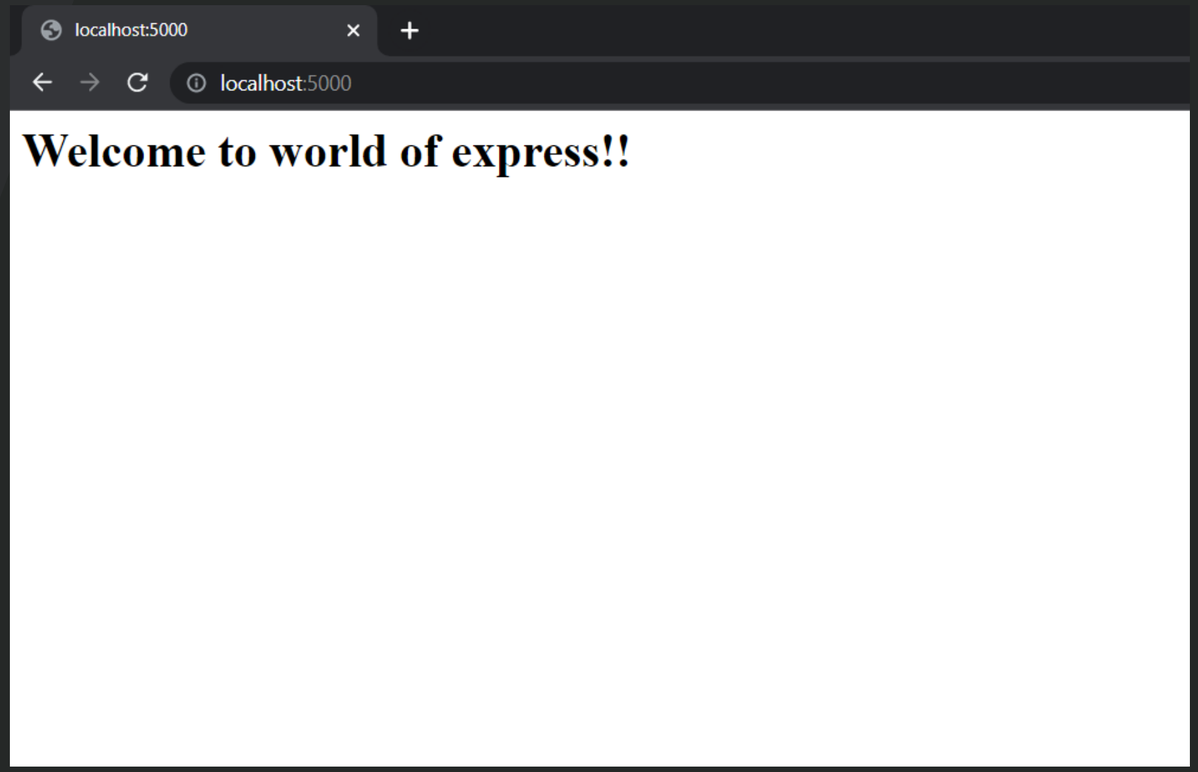
Express JS

What is ExpressJS?

It is framework of Node.js web server simplify routing and middleware

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Let's build our first website in ExpressJS !!



Express app objects

- app.all
- app.use
- app.get
- app.post
- app.delete
- app.put
- app.listen

WLUG
COMMUNITY | KNOWLEDGE | SHARE

What are Routers?

Routers

- What is the servers response to in an incoming request?
- How an application's endpoints (URIs) respond to client request?
- A router instance is a complete middleware and routing system

Router Objects

You can group routes together using routers which can be used for organisation and to apply middleware to a specific group of routes

```
// create the router object
const router = express.Router()
// register it with the application for routes with a certain prefix
app.use("/prefex", router)
```

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Routes

Created by using one of the following method

- .all
- .post
- .get
- .delete
- .put
- .patch

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Routes

All of these functions take two arguments:

- The endpoint
- **Controller** or **Route Handler** function that takes req and res as arguments

Routes

```
// writing pass an anonymous function
app.get("/endpoint", (req, res) =>  {
  res.send("The Response")
});

// using a named function
function routeHandler(req, res){
  res.send("the response")
}
app.get("/endpoint", routeHandler);
```

Request Object

Represents the data from the incoming request and is passed to all middleware and route handlers

- req.headers
- req.params
- req.query
- req.body
- req.method

Response Object

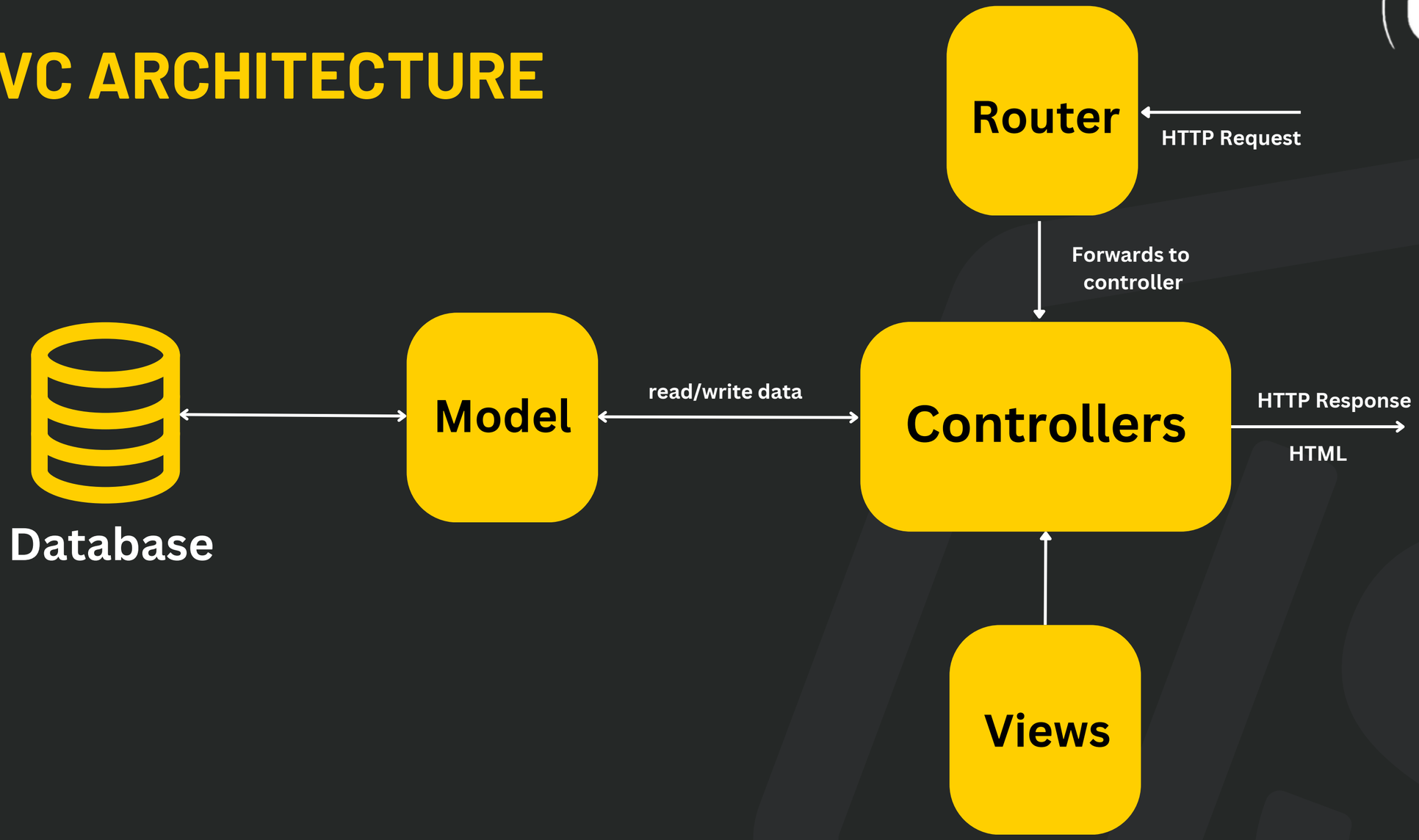
It is an object that is used to help author the response to the request

- res.send
- res.json
- res.render

TECHNOTWEET

WLUG
COMMUNITY | KNOWLEDGE | SHARE

MVC ARCHITECTURE



What is Database?

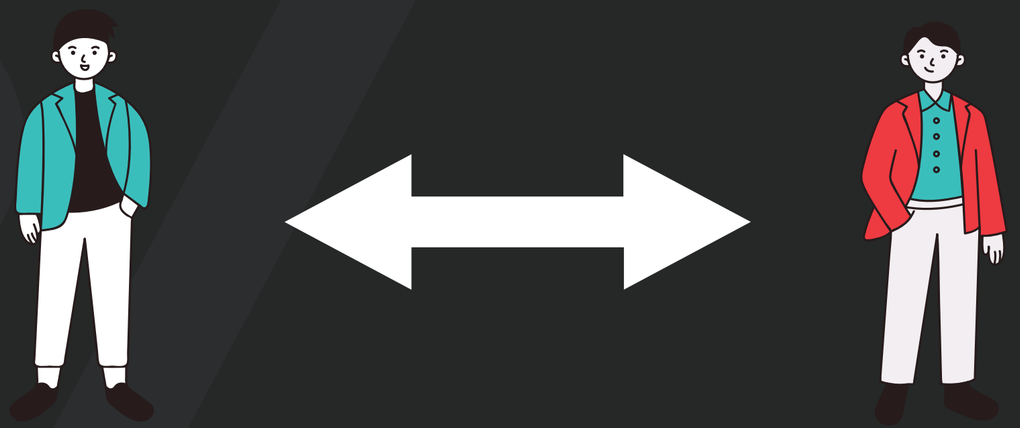
SQL vs NoSQL

- Relational Database Management System (RDBMS)
- Fixed or static or predefined schema
- Not suitable for hierarchical data storage.
- Vertically Scalable
- Follows ACID property
- MySQL, PostgreSQL, Oracle

- Non-relational or distributed database system
- Dynamic schema
- Best suited for hierarchical data storage
- Horizontally Scalable
- Follows CAP Theorem
- MongoDB, GraphQL, HBase

WLUG
COMMUNITY | KNOWLEDGE | SHARE

CAP



A

B

MONGODB

- Based on a non-relational document model
- NoSQL database
- Differs from relational DBs such as MySQL, Oracle
- Default Port number is 27017

mongoDB

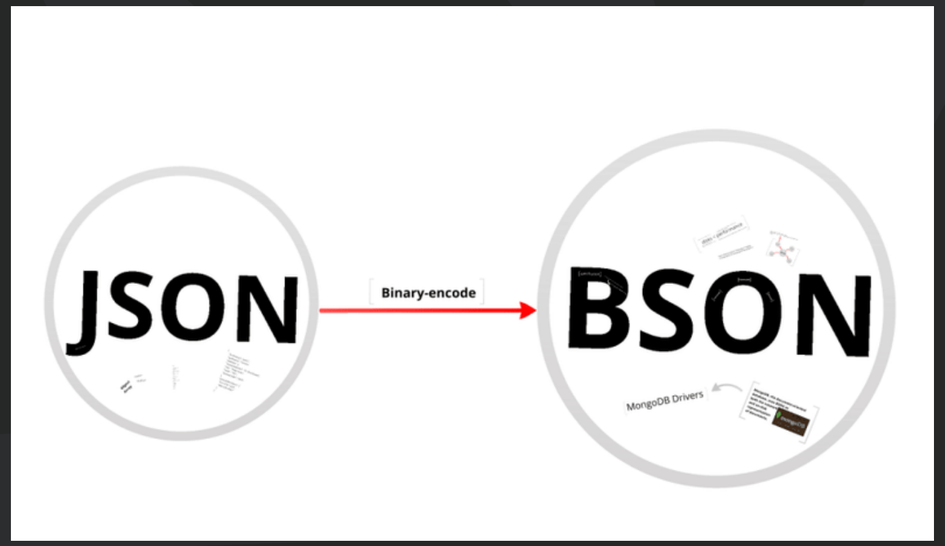
Collection

- Group of MongoDB documents
- Equivalent to Tables in Relational Databases
- Stores documents in BSON format

```
db.users.insertOne(      <------ Collection
  {
    name: "sue",         <----- field:value
    age: 26,             <----- field:value
    status: "pending"    <----- field:value
  }
)
```

WLUG
COMMUNITY | KNOWLEDGE | SHARE

JSON & BSON



JSON

```
{
  "array": [
    1,
    2,
    3
  ],
  "boolean": true,
  "null": null,
  "number": 123,
  "object": {
    "a": "b",
    "c": "d",
    "e": "f"
  },
  "string": "Hello World"
}
```

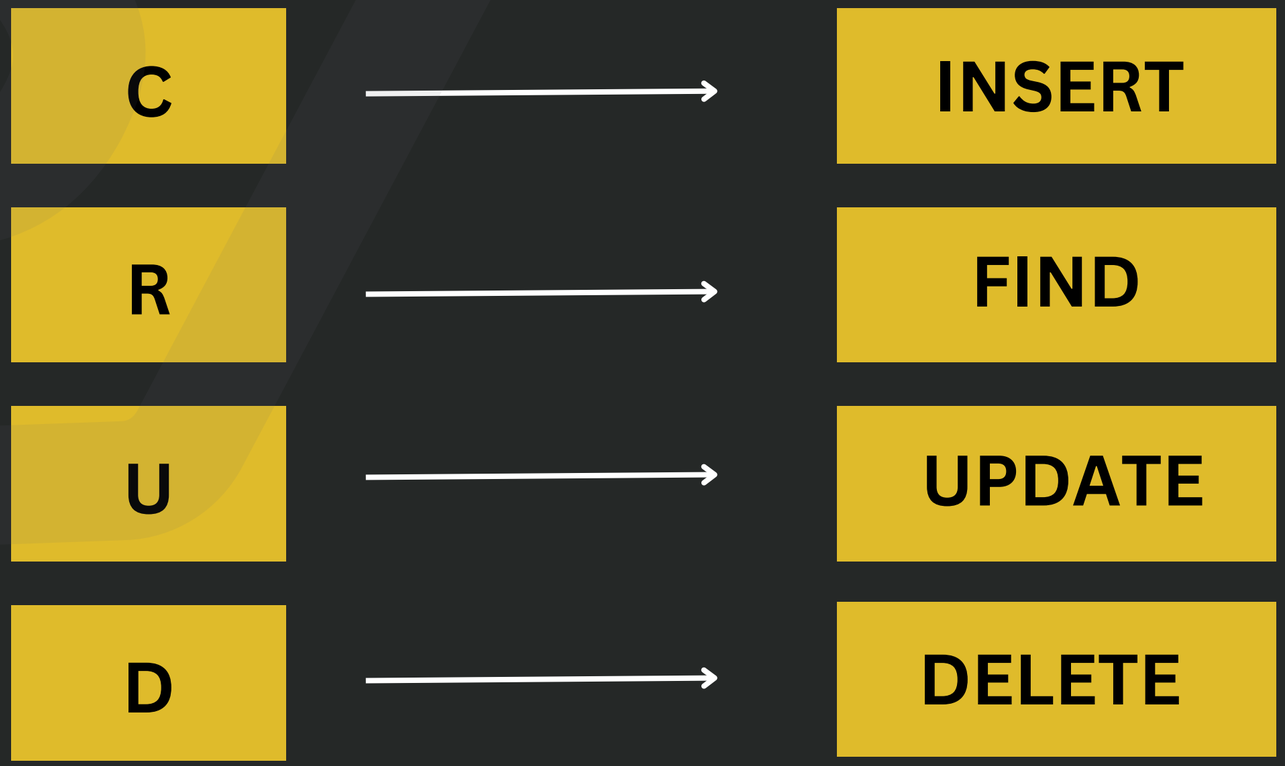
BSON

```
{ 01010100
  11101011
  10101110
  01010101 }
```

WLUG
COMMUNITY | KNOWLEDGE | SHARE

MongoDB Crud Operations

- CRUD operations are create, read, update, and delete documents



TECHNOTWEET

WLUG
COMMUNITY | KNOWLEDGE | SHARE

MongoDB Compass Installation

Mongoose

Mongoose:

- ODM (Object Data Modeling) library for MongoDB
- Define data models using a schema-based approach
  - Data Model
  - Schema

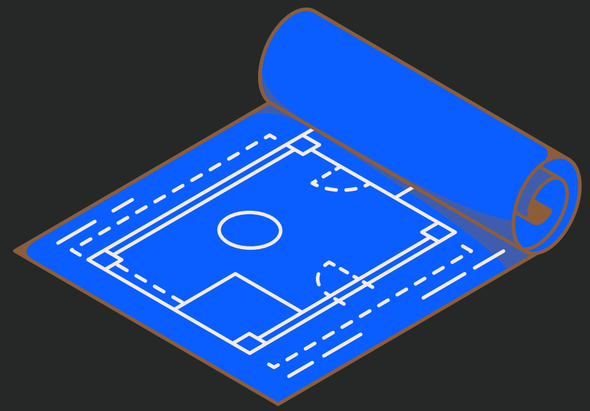
Installing Mongoose:

```
npm install mongoose
```

Schema:

- Structure of Collection Documents
- mongoose.Schema() method

- Different fields in Mongoose
  - type: String, Number, Boolean, Date
  - default
  - required
  - unique



Schema:

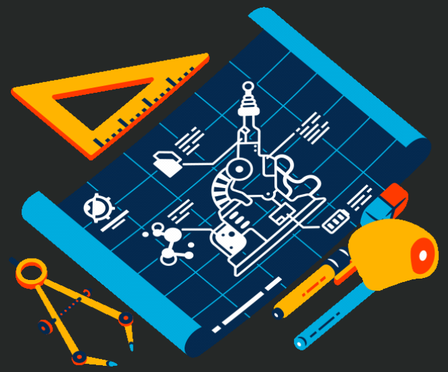
```
const mongoose=require('mongoose')
const BlogPostSchema = new mongoose.Schema({
    title: {
        type: String,
        required: true
    },
    author: {
        type: String,
        required: true
    },
    description: {
        type: String,
        required: true
    },
    content: {
        type: String,
        required: true
    },
    date: {
        type: Date,
        default: Date.now
    }
});
const BlogPost=mongoose.model('BlogPost',BlogPostSchema)
module.exports=BlogPost
```

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Model:

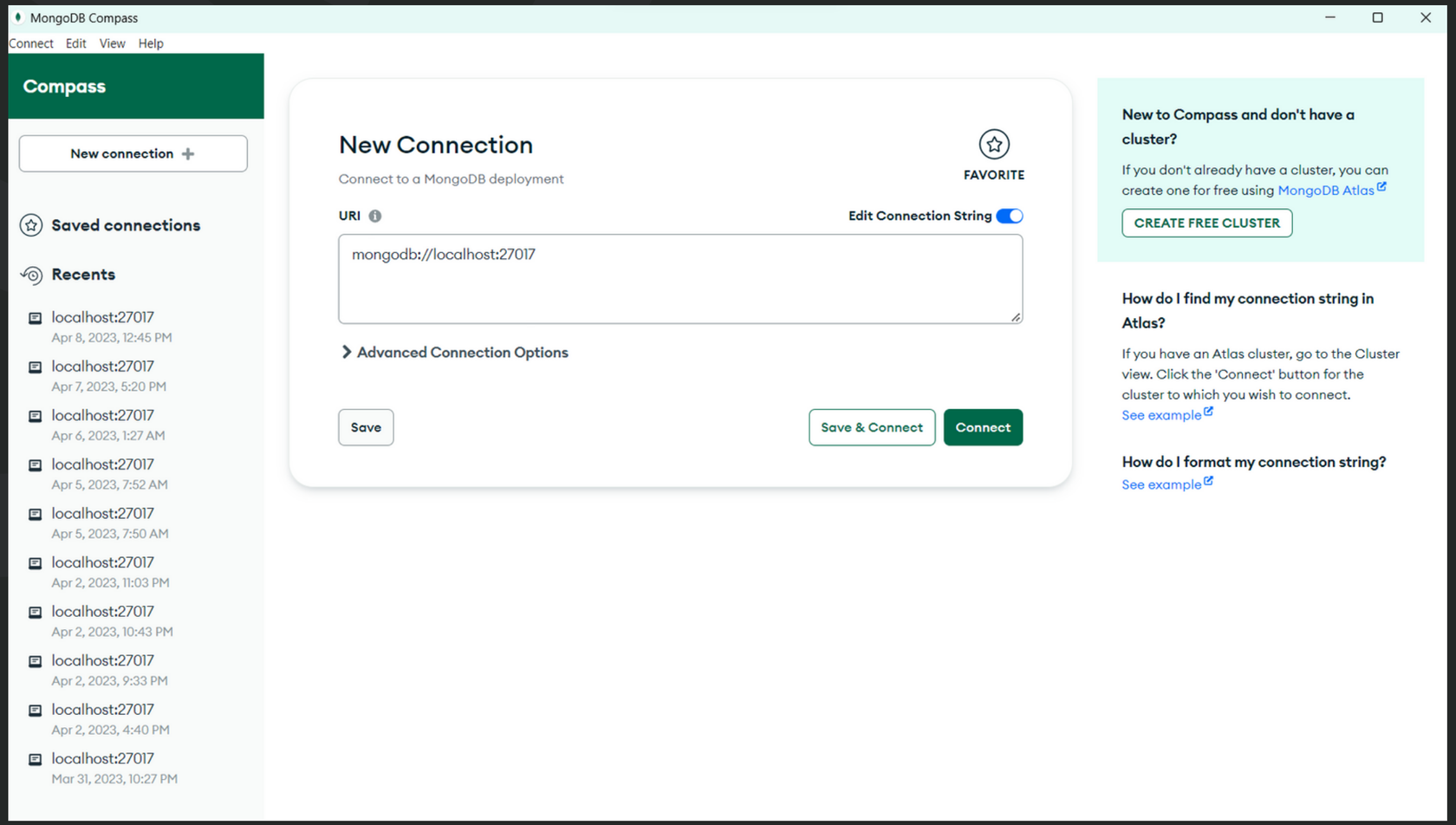
- Interface for manipulating documents in that collection
- We can create a model by calling the **mongoose.model()** method

```
const newPost=mongoose.model('NewPost',BlogPostSchema)
```



WLUG
COMMUNITY | KNOWLEDGE | SHARE

Mongodb Compass:



WLUG
COMMUNITY | KNOWLEDGE | SHARE

Connecting To DB Using Mongoose :

- We connect to MongoDB by calling the **mongoose.connect()** method

```
const mongoose=require('mongoose')
    const func = async() => {

        await mongoose.connect('mongodb://0.0.0.0:27017/Mydatabase').then(()=>{
            console.log("connected");
        }).catch((err)=>{
            console.log(err);
        })
    }
```

TECHNOTWEET

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Let's Use The Mongoose !

WLUG
COMMUNITY | KNOWLEDGE | SHARE

What are Middlewares?

Middlewares

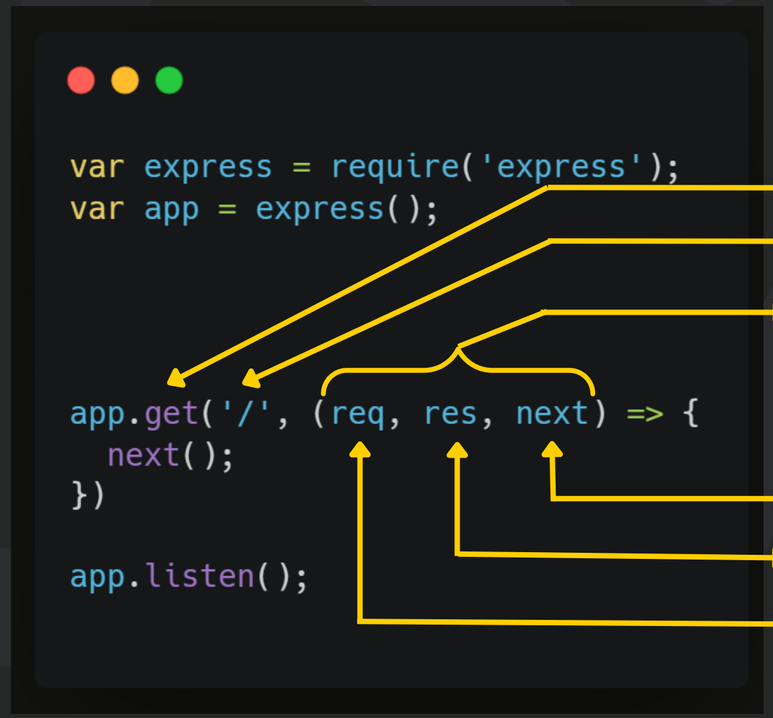
- Have access to the request object (req), the response object (res), and the next function in the application's request-response cycle
- Executed during the request-response cycle in ExpressJS
- Executed in the order in which they are defined

Middleware functions in ExpressJS:

- Execute any code
- Make changes to the request and the response objects
- End the request-response cycle
- Call the next middleware in the stack

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Elements of a Middleware function:



HTTP method for which the middleware function applies

Path (route) for which the middleware function applies.

The middleware function

Callback argument to the middleware function, called "next" by convention

HTTP response argument to the middleware function, called "res" by convention

HTTP request argument to the middleware function, called "req" by convention

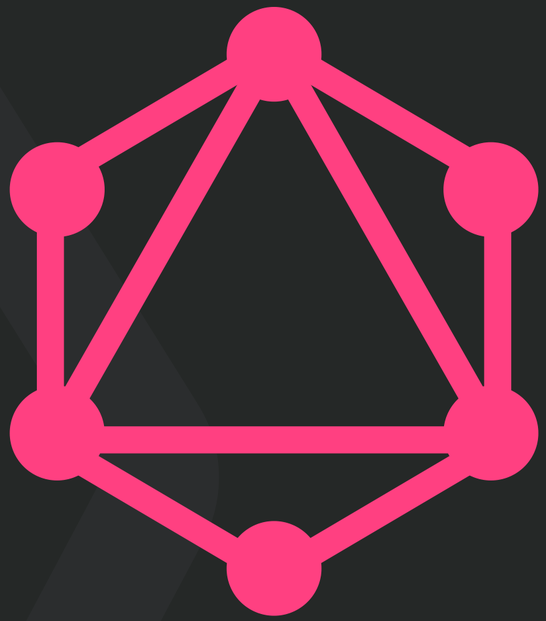
WLUG
COMMUNITY | KNOWLEDGE | SHARE

Let's create Controllers!

Graphql

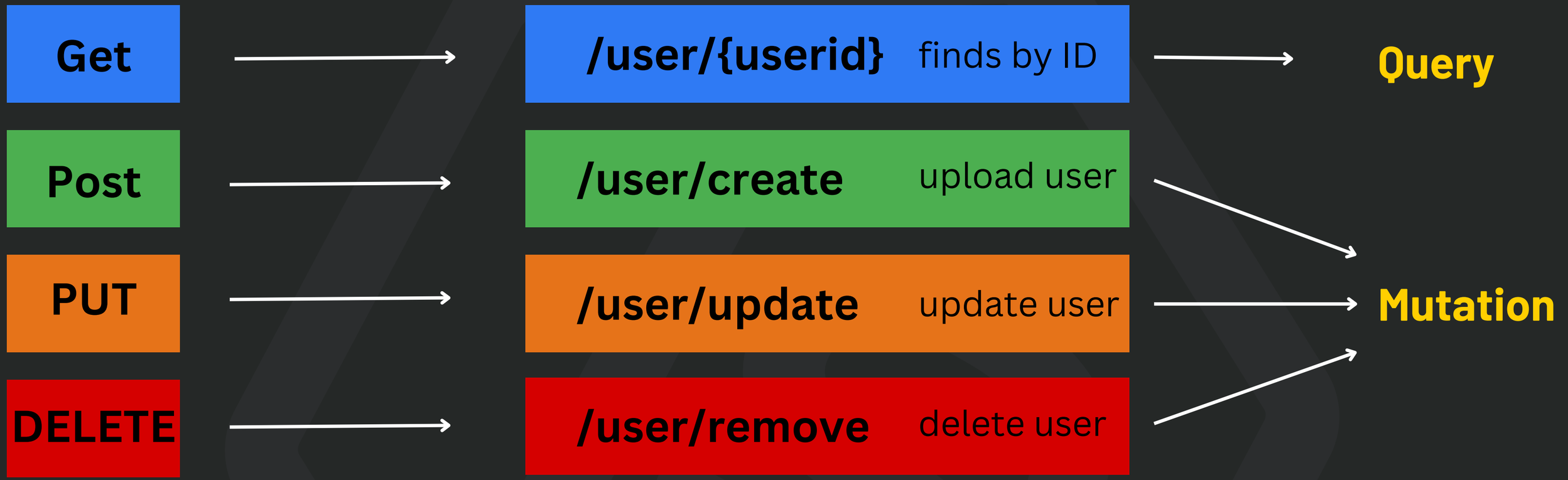
Graph Query Language

- A query language for APIs
- Unlike REST APIs, multiple queries per request
- There is no need to remember hunderds of routes



Common misconception:

Graphql is not a database, don't compare it with SQL


Get
/user/{userid} finds by ID
Query
Post
/user/create upload user
PUT
/user/update update user
Mutation
DELETE
/user/remove delete user

WLUG
COMMUNITY | KNOWLEDGE | SHARE

Let's Deploy!!

```
res.send('Thank You!');
```

WLUG
COMMUNITY | KNOWLEDGE | SHARE

console.log('Thank You!');