Modul 3

Intro to Express

Outline

- What is Express?
- Advantages of Express
- Express core features

Definition

“Express is a minimal Node.js web application framework that provides a robust set of features for web and mobile applications.”

- Express assists in creating server-side applications faster and smarter
- Simple, minimalist, flexible, and scalable are some of the characteristics of Express js.
- What Express is to Node js is what Bootstrap is to HTML and CSS.
- Express js can cut your coding time in half.

- Express is one of the most used framework for developing node js applications.
- Express js makes it very easy to handle multiple types of requests such as GET, POST, PUT, DELETE.
- By using Express, we gain access various open source softwares that solves commons problems of development.
- Unopinionated.
  - Devs are free to pick whatever libraries they want
  - Provides flexibility and capability to highly customize their project.
  - Insert almost any compatible middleware.
  - Structure the app in one or multiple files.

Some of Express' Core Features

- Allows to set up middlewares to respond to HTTP Requests.
- Defines routes which then will perform different functions.
- Dynamically render HTML pages based on passing arguments to templates.

Middleware?

- Middleware is that acts as a bridge between an operating system or a database and an application, especially on a network.
  - Being “in the middle” of 2 programs hence the name middleware.

- Middleware features include
  - Database
  - Deployment
  - Virtualization
  - Authentication
  - Etc

Middleware vs API

- While both are intermediaries of 2 programs
  - Middleware traditionally means a bridge between 2 programs (back and forth)
  - API makes thing more omnidirectional
    - Through middlewares, one API can call multiple programs.
    - API gives access to the functionality of a middleware.
- By definition, an API is also a middleware due to its position in between the database and the client.
- In our case, Middlewares are programs we need to connect our API with other external programs.

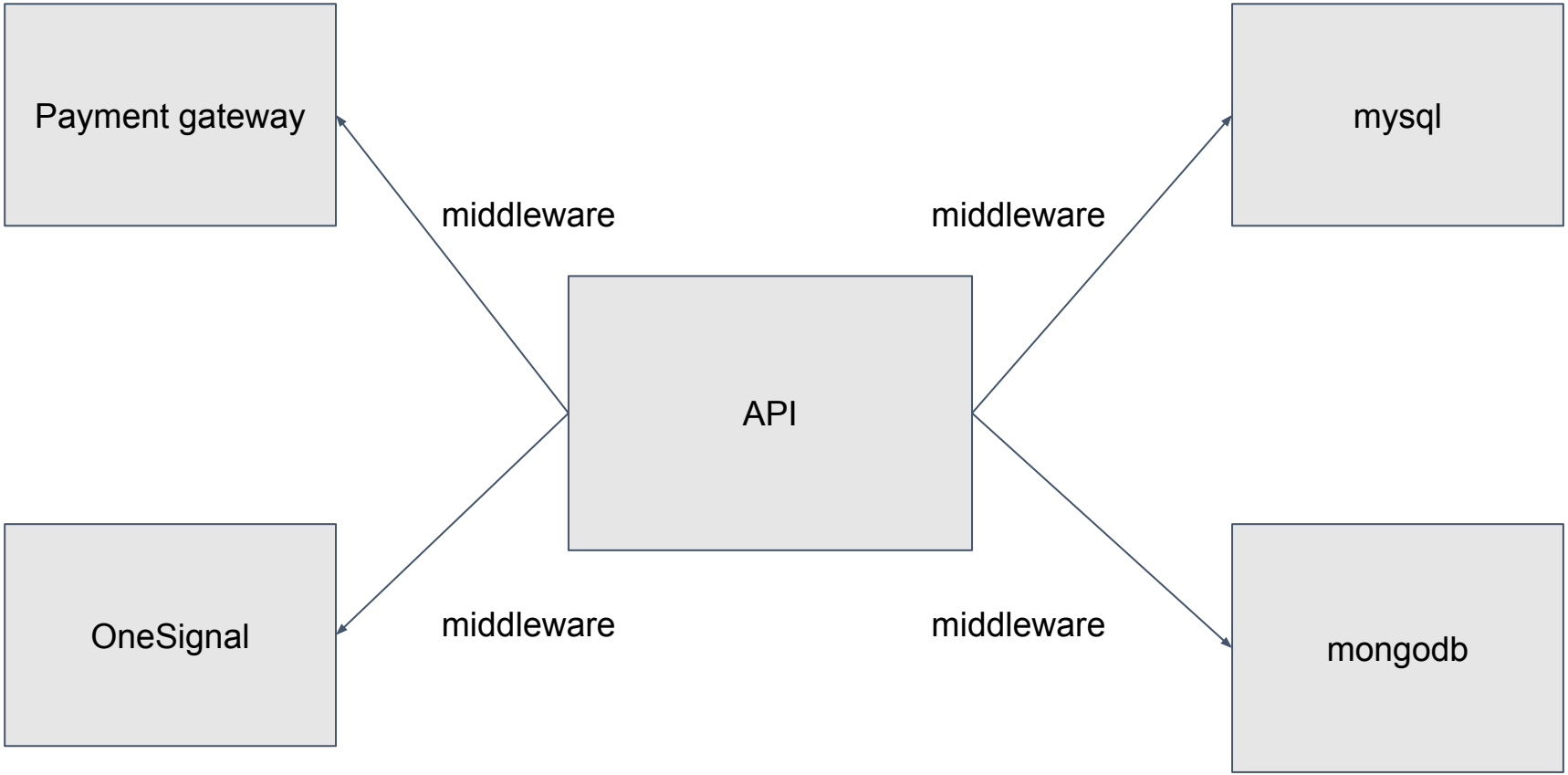
```
const express = require("express");
const app = express();
const mysql = require('mysql');

// mysql is a middleware in our API
// Allowing our API to connect to a mysql database
// The code below creates a connection to mysql
const db = mysql.createConnection({
  host: localhost,
  user: "lianeddy",
  password: "*******",
  database: "my_database",
  port: 3306,
});

app.get('/get' , (req,res) => {
  // The function below is provided by the middleware
  // Allowing us to fetch data from the mysql database
  db.query("SELECT * FROM products", (err, data) => {
    if(err) res.status(500).send(err);

    res.status(200).send(data);
  })
};
...
```

Purwadhika
Startup and Coding School



Purwadhika
Startup and Coding School

Middleware Functions

- Middleware functions are functions that have access to the request object (obj) and the response object.
- Middleware functions can perform the following:
  - Execute code
  - Make changes to the request and response
  - End request - response cycle
  - Call the next middleware in the stack

```
app.get('/', (req, res, next) => {
    next();
});
```

```
const express = require('express');
const app = express();
const port = 2000;

const getRequestTime = (req, res, next) => {
  req.time = Date.now();
  next();
};

app.get('/', getRequestTime, (req,res) => {
  res.send(`Hello your call time is at ${req.time}`);
});

app.listen(2000, () => console.log(`API listening at port ${port}`);
```

```
const express = require('express');
const app = express();
const port = 2000;

const authenticate = (req, res, next) => {
  if(req.query.id === req.params.id){
    next();
  }
  else{
    return res.status(401).send('Not Authorized');

app.get('/', authenticate, (req,res) => {
  // database manipulation
});

app.listen(2000, () => console.log(`API listening at port ${port}`);
```