Period-2 Node, Express with TypeScript, JavaScript Backend Testing, MongoDB (and Geo-location)

Explain Pros & Cons in using Node.js + Express to implement your Backend compared to a strategy using, for example, Java/JAX-RS/Tomcat

SVAR: Fordelen er, at der kan bruges ét sprog til det hele, nemlig JavaScript. Det er et populært sprog med mange biblioteker, og det er nemt at finde information og hjælp online.

Desuden er det nemt og hurtigt at sætte op. Der kan køres en express generator, og så genereres et projekt med en fordelagtig mappestruktur.

Explain the difference between *Debug outputs* and *Application Logging*. What’s wrong with console.log(..) statements in our backend code?

SVAR: Winston, som er et application logging værktøj, giver flere muligheder end blot at skrive console.log(). Ved brug af Winston, kan loggen f.eks. gemmes i en database. Loggin overvåger driften og kan bruges til at opdage fejl i applikationen.

Debugging bruges til at finde frem til fejl, når man allerede ved, at noget i koden ikke er, som det skal være.

Console.log() er det output, der ”udskrives”. Det er bedre at loggetil en database eller en fil, så logningen kan tilgås.

Demonstrate a system using application logging and environment controlled debug statements.

SVAR:

Application logging: Findes I startcode/arc/app.ts.

Environment controlled debug statements: NODE\_ENV kan ændres til enten production eller development. Førstnævnte logger til combined.logs, og sidstnævnte logger til konsollen.

Explain, using relevant examples, concepts related to testing a REST-API using Node/JavaScript/Typescript + relevant packages

SVAR:

**Chai:** Et assertion bibliotek, der ofte bruges sammen med Mocha. Metoder, der sammenligner outputtet af funktioner med forventede resultater. Eksempel: expect(exampleArray).to.have.lengthOf(3);

**Mocha:** funktionerne eksekveres i en specifik rækkefølge, og resultaterne logges til terminalen. Der bruges keywordsene ”describe” og ”it”

**Supertest:** Bruges til at teste http endpoints.

**Nock:**

**Explain a setup for Express/Node/Test/Mongo-DB/GraphQL development with Typescript, and how it handles "secret values",  debug, debug-outputs, application logging and testing.**

**SVAR:**

Secret values: I .env-filen lægges de hemmelige variabler, og .env tilføjes til .gitignore. På denne måde eksponeres de ikke på GitHub.

**Explain a setup for Express/Node/Test/Mongo-DB/GraphQL development with Typescript. Focus on how it uses Mongo-DB (how secret values are handled, how connections (production or test) are passed on to relevant places in code, and if use, how authentication and authorization is handled**

**SVAR:**

Explain, preferably using an example, how you have deployed your node/Express applications, and which of the Express Production best practices you have followed.

Explain possible steps to deploy many node/Express servers on the same droplet, how to deploy the code and how to ensure servers will continue to operate, even after a droplet restart.

Explain, your chosen strategy to deploy a Node/Express application including how to solve the following deployment problems:

* Ensure that you Node-process restarts after a (potential) exception that closed the application
* Ensure that you Node-process restarts after a server (Ubuntu) restart
* Ensure that you can run “many” node-applications on a single droplet on the same port (80)

Explain, using relevant examples, the Express concept; middleware.

SVAR: Middleware funktioner er funktioner, der har adgang til request og response objektet og next-funktionen. Middleware kommer imellem request og response cyklussen.

Next-parametret er en del af Middleware-funktioner, fordi den passerer requestet videre til næste funktion i kæden. Sidste led i funktionen, har ingen next.

Explain, conceptually and preferably also with some code, how middleware can be used to handle problems like logging, authentication, cors and more.

SVAR:

Authentication: src/middleware/basic-auth

Logger: src/middleware/logger og simpleLogger

Cors:

import express from "express";

const app = express();

export default app.use((req, res, next) => {

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

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

    next();

});

**Explain, using relevant examples, your strategy for implementing a REST-API with Node/Express  + TypeScript** and demonstrate how you have tested the API.

SVAR:

App.get bruges til at skabe ruter med.

Explain, using relevant examples, how to test JavaScript/Typescript Backend Code, relevant packages (Mocha, Chai etc.) and how to test asynchronous code.

SVAR: Se test-klasserne. De forskellige pakkers formål er forklaret længere oppe.

NoSQL and MongoDB

*Explain*, generally, what is meant by a NoSQL database.

SVAR: En NoSQL db gemmer dataen anderledes end SQL db’s. Der er ingen tabeller, så dataen gemmes, uden at den skal passe ind i et skema.

***Explain* Pros & Cons in using a NoSQL database like MongoDB as your data store, compared to a traditional Relational SQL Database like MySQL.**

**SVAR:**

**Pros:**

* Fleksibel datamodellering. Det er nemt at gemme og kombinere forskellige typer af data.
* High performance
* Scalable

**Cons:**

* Ikke så modent som SQL. SQL er på et mere avanceret udviklingsstadie. Det gør det sværere at finde hjælp og support
* SQL er mere generaliseret og kan bruges til næsten alt. NoSQL er bedst til specifikke cases.

*Explain* about indexes in MongoDB, how to create them, and *demonstrate* how you have used them.

SVAR: Indexes er en speciel datastruktur, der indeholder daata fra et par felter. Indexes gør det hurtigere at søge i databasen. I stedet for at søge i hele databasen, søges der kun på indexes med de få relevante felter.

Her er et eksempel. Det gør det hurtigere at søge fordi indexet vil blive brugt i søgningen. Det er vigtgit at lave index på det felt, der vil blive søgt mest på.

db.studentdata.createIndex({student\_name: 1})

*Explain*, *using your own code* examples, how you have used some of MongoDB's "special" indexes like *TTL* and *2dsphere and perhaps also the Unique Index.*

*SVAR: makeTestFriends.ts, linje 13*

***Demonstrate*, using your own code samples, how to perform all CRUD operations on a MongoDB**

**SVAR:** FriendFacade.ts og FriendRoutes.ts

Demonstrate how you have set up sample data for your application testing

SVAR:

Explain the purpose of mocha, chai, supertest and nock, which you should have used for your testing

SVAR:

**Chai:** Et assertion bibliotek, der ofte bruges sammen med Mocha. Metoder, der sammenligner outputtet af funktioner med forventede resultater. Eksempel: expect(exampleArray).to.have.lengthOf(3);

**Mocha:** funktionerne eksekveres i en specifik rækkefølge, og resultaterne logges til terminalen. Der bruges keywordsene ”describe” og ”it”

**Supertest:** Bruges til at teste http endpoints.

Explain, using a sufficient example, how to mock and test endpoints that relies on data fetched from external endpoints

SVAR:

Explain, using a sufficient example a strategy for how to test a REST API. If your system includes authentication and roles explain how you test this part.

SVAR:

Endpoints testes fra klassen friendEndpointTest.ts

Der bruges en inMemory database fra Mongo, så der ikke interageres med den rigtige database, og data ikke kan ændres.

*Explain*, using a *relevant example*, a full JavaScript backend including relevant test cases to test the REST-API (not on the production database)

SVAR: