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

Note: This description is too big for a single exam-question. It will be divided up into separate questions for the exam

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

Største fordel ved Node og Express som jeg ser det, er hvor hurtigt man kan komme i gang, på meget få linjer kode har man et api op og køre.

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

Med Debug har vi mulighed for at bestemme hvornår vi vil udskrive til konsollen, ud fra i hvilken miljø vi arbejder i, man kan hurtig komme til at glemme et console.log statement, som så vil blive vist ude hos brugeren i produktion.

Med Application logging er det på samme måde, vi vælger at logge til en fil hvis vi er i produktion, men ikke hvis vi er i development.

Så alt vi vil have logget bruger vi vores logger til, i produktion vil dette blive logget til en fil.

Debug vil give os beskeder i konsollen men aldrig logge noget til filer.

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

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/src/bin/www.ts

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/src/app.ts

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

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/test/friendEndpointsTest.ts

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.

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/src/app.ts

Vores secret values ligger vi i vores .ENV fil.

Vi bruger Winston til vores logging, hvor vi kan bestemme de forskellige ”levels” af fejl, info, severe osv.Winston kan skrive til vores log filer, udfra om vi er i produktion eller dev.

Vi sætter vores logger til app objectet, så vi kan hente loggeren ned ved app.get(”logger)

Morgan bruger vi til at implementere vores winston logging som middleware. På min startkode har jeg sat morgan op til at kigge på alt på app

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

.env filen indeholder vores mongodb connection credentials

I dbconnector ligger vores connections til MongoDB, vi bruger en som connecter til vores in memory-db, og en til vores ”rigtige” database.

Når vi tester, i vores before all, så sætter vi vores db connection op til at bruge vores memory server, og derefter sætter vores db på application objectet, når vi instansiere vores facade i friendRoutes, så hiver vi db ud af objektet som fortæller hvilken DB vi bruger.

Authentication middleware sætter vi på router efter ”/friends/” så alle endpoints som ikke har noget med at oprette en ny friend skal autentikere sig, under alle admin tasks sikre vi os at brugeren har den rigtige rolle som er ”admin”

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.

Se I app.ts LINK her, eller AUTH middleware i friendroutes. LINK HER

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

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/src/app.ts

https://github.com/Pelle-pr/fullstack-typesript-startcode/tree/main/src/middleware

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

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/src/routes/FriendsRoutes.ts

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/test/friendEndpointsTest.ts

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

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/src/facade/friendFacade.ts

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/test/friendFacadeTest.ts

NoSQL and MongoDB

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

Structure: tables, documents (MongoDB), graphs

Kører med Key value store, dvs vi har en key som f.eks en stregkode på et produkt, og en value som kunne være et json objekt, dette objekt kan vi tilføje flere fields til hvis vi vil, dette er mere problematisk i en relational sql database.

Storage: vores key value bliver hashed og lagt ud på flere partitioner på serveren.

Scale: Meget let tilføje flere partitioner (horizontal)

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

MySql databaser er vertikal skalerbare, det vil sige at man kan sørge for at ens sql server f.eks bliver opgraderet med mere memory, CPU kraft etc.

MySql databaser bruger tabels med fixed rows og colums, i mens NoSQL bruger documents, hvor man i princippet kan smide alt ind man har lyst til, hvis man skal bruge en ekstra ”column / field” så smider man det bare ind.

NoSql databaser er horizontal skalerbare, det vil sige man hurtigt bare lave flere af dem, så man kan klare mere traffic ved hjælp af ”sharding”

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

this.friendCollection.createIndex({ email: 1 }, { unique: true });

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/src/facade/friendFacade.ts

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.*

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

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/src/facade/friendFacade.ts

Eller p2/day3/

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

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/test/friendEndpointsTest.ts

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/test/friendFacadeTest.ts

(BeforeEach)

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

Mocha er vores test framework.

Chai er vores expectation biblotek, som vi bruger i stedet for nodes standard assert.

Supertest bruger vi til at køre en test server til vores endpoints, og lave hurtige simple http requests.

Nock bruger vi til at intercepte et eksternt http kald, da vi ikke kan være sikre på at vi altid får det samme return, så kan vi lave en "Mock" af et response, som vi kan teste på.

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

https://github.com/Pelle-pr/fullstack-typesript-startcode/tree/main/playground/usingNockNameInfo

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.

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/test/friendEndpointsTest.ts

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

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/test/friendEndpointsTest.ts

https://github.com/Pelle-pr/fullstack-typesript-startcode/blob/main/test/friendFacadeTest.ts

Geo-location and Geojson (Period-4)

Explain and demonstrate basic Geo-JSON, involving as a minimum, Points and Polygons

Explain and demonstrate ways to create Geo-JSON test data

Explain the typical order of longitude and latitude used by Server-Side APIs and Client-Side APIs

Explain and demonstrate a REST API that implements geo-features, using a relevant geo-library and plain JavaScript

Explain and demonstrate a REST API that implements geo-features, using Mongodb's geospatial queries and indexes.

Explain and demonstrate how you have tested the gameFacade and gameAPI for the game-related parts of the period exercises

This will come in period-5

Explain and demonstrate a React Native Client that uses geo-components (Location, MapView, etc.)

Explain and demonstrate both server and client-side, of the geo-related parts of your implementation of the ongoing semester case.