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**

Min besvarelse:  
Når man søger om dette emne på internettet er der mange delte meninger.   
  
Pros: Hurtigere at arbejde med (mindre kode til samme funktionalitet). Konstant i udvikling. Samme programmeringssprog til front- og backend, hvilket gør det nemmere at udvikle på samme sider (kan også gøres med Java fx med JSP sider, men det er ikke så ”moderne” og brugervenligt for slutbrugeren).

Cons: Java+Tomcat setup er mere udbredt i enterprise level løsninger, med mange flere år på bagen. Og derfor er det mere gennemtestet en Node.js. Node.js er et framework bygget på JavaScript, så du får ikke den ”rene vare” ligesom med Java.

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

Min besvarelse:  
Debug er et værktøj man bruger aktivt som udvikler mens man er i gang med at udvikle, for lettere at kunne løse fejl *som man har identificeret*.

Applicationlogging er et værktøj vi som udvikler bruger til at overvåge driften af vores applikation, for *opdage fejl* i vores applikation.

Console.log() duer ikke i backend, det jo blot er beskeder der bliver vist i konsollen i backend, og derfor ikke noget vi opdager som udgangspunkt. Derfor vil vi gerne logge til en fil, database eller andre ’persistente’ steder.

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

Min besvarelse:  
Se projektet i ’P2 code examples’ -> ’startcode’.

Her bestemmer en miljøvariabel i .env filen (’NODE\_ENV’) hvilken type logging vi kører. Hvis applikationen er i produktion, vil vi logge til ’logs’ -> ’combined.log’. Hvis applikationen er i dev, vil vi logge til konsollen.

I eksemplet har jeg valgt at bruge ’winston’ til at logge. Koden kan ses i ’middleware’ -> ’logger.ts’.

I forhold til debugging har vi i vores ’package.json’ følgende linje kode:   
”dev:debug": "nodemon --inspect src/bin/www.ts”  
Som giver os mulighed for at køre applikationen i debug måde, så vi eksekverer koden linje for linje.  
  
Derudover har vi i .env filen denne linje kode:  
”DEBUG=www,app,setup-friend-testdata,friend-routes”  
Som giver os mere information i konsollen når de definerede filer bliver eksekveret.

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

Min besvarelse:  
Se ’P2 code examples’ -> ’startcode’ -> ’test’ -> ’friendEndpointTest.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.**

**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**

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.**

Min besvarelse:  
Se ’P2 code examples’ -> ’startcode’ -> ’app.ts’, hvor vi bruger middleware med app.use(xx) og app.get(xx).

Når vi bruger middleware kan vi læse og skrive til ’request’ og ’response’ requestet, hvilket gør det muligt at authenticate, authorize, logge, validere, omdirigere og en masse andre ting.

Rækkefølgen på middleware i koden bestemmer hvilket middleware vi rammer først. Hvis vi bruger next() metoden i noget middleware ryger vi videre til det næste middleware.

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

Min besvarelse:  
Se ’P2 code examples’ -> ’startcode’ -> ’app.ts’.

I linjen:  
”app.use("/api/friends", cors(), friendsRoutes)”  
Bruger vi middleware til at sørge for at der for alle requests med addressen "/api/friends" bliver bruge modulet ’friendRoutes’. På den måde slipper vi for at skrive "/api/friends" foran alle vores api metoder i ’friendRoutes’.  
Derudover sender vi cors() med til ’friendRoutes’.

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

Min besvarelse:  
Se ’P2 code examples’ -> ’startcode’ -> ’app.ts’.

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

Min besvarelse:  
Se ’P2 code examples’ -> ’startcode’ -> ’test’ -> ’friendFacadeTest.ts’ for eksempler på test af backend koden med brug af Mocha (test framework) og Chai (assertions til Mocha).

Vi tester asynkron kode ved at lave mock responses til vores requests. Til dette kan vi bruge pakken ’supertest’ som kan intercepte API kald og returnere en custom made ’mock’ response.

#### NoSQL and MongoDB

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

Min besvarelse:  
Det betyder at databasen ikke er relationel.   
En SQL database er relationel, så vi organiserer data i tabeller bestående af kolonner og rækker, hvor hver række har en unik primary key. Og der er en logisk forbindelse nogle tabeller, som er etableret på regler. Der er også regler for hvordan typen af data i hver kolonne skal være, samt hvor mange kolonner der skal være i hver række.   
Disse regler gælder ikke i en NoSQL database.

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

Min besvarelse:  
Pros: Mere fleksibel. Horisontalt skalérbare (SQL er kun vertikalt skalerbart). Bedst til ustruktureret data.

Cons: Større risiko for fejl og sværere at dokumentere. Best til struktureret data.

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

Min besvarelse:  
We have note used indexes?

**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.**

Min besvarelse:  
We have note used indexes?

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

Min besvarelse:  
Se ’P2 code examples’ -> ’mongodb’ for eksempel på alle CRUD operationer.

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

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

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

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.

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

#### 

#### 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.