

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

- Node.js Pros:
 1. Når koden skal deployes på en server, skal den ikke kompileres om til en .war-fil for at kunne køre, da det i forvejen er et maskinsprog.
 2. Ikke tungt at køre i forhold til Java
 3. Der kommer mange opdateringer, den bliver hele tiden holdt opdateret for ændringer
- Node.js Cons:
 1. Der kommer mange opdateringer, skal hele tiden holdes opdateret
 2. Hvis et library bliver slettet fra NPM virker programmet/applikationen ikke længere

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

- `Console.log()` er blokerende for stacken.
- Debug outputs bruges ligesom `console.log()`, men da `console.log()` er blokerende, bruger man `debug()`, da disse ignoreres i produktion. `ApplicationLogging` logger alt hvad der sker i app'en. Det kan være hvilke api'er der bliver tilgået og af hvem, det kan være fejlbeskeder osv.

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

- Periode 2 projekt fra uge 1-3

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

- `UserEndpointTest` i `gameProject`
- <https://www.npmjs.com/package/chai-as-promised>
- <https://www.chaijs.com/plugins/chai-as-promised/>
- **Chai** is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework.
- **Mocha** is a feature-rich JavaScript test framework running on [Node.js](https://nodejs.org/) and in the browser, making asynchronous testing *simple* and *fun*. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.

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

- Hvis jeg forstår spørgsmålet korrekt, så har vi en .env fil der indeholder de mere følsomme oplysninger. Dette er blandt andet et link til vores MongoDB Cluster(uri), som vi ved hjælp af nogle importes kan hente ud og anvende i vores kode. Denne .env bliver ikke skubbet med op på github og er derfor kun synlig for os selv. Det er muligt at slå authentication til og fra i .env filen, samt at toggle debug statements til og fra.

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

- <https://expressdemo.tunoc.codes/> - Tobias link til server app
- <https://expressjs.com/en/advanced/best-practice-performance.html>
- Alt kode er async
 - Non blocking
- logger alle endpoints og errors
 - Express-Winston
 - (Logger, Error Log)
- Håndtere exception/anvender middleware
 - Endpoint error ("This endpoint doesn't exist" - 404)
 - Error format, Custom APIerror(Error code, message)
- Sat NODE_ENV til produktion
 - Oprette en .env fil efter cloning til droplet.
 - Sætte de korrekte env variable ind i .env folderen. (Connection, port, DB name osv.)
- Digitalocean er sat til selv at starte
 - "pm2 startup" provides us with a command to auto start out application
 - Så genstarter vi dropletten
 - pm2 start ./build/app.js --watch --ignore-watch="node_modules"

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.

- Brug subDomains for at have flere forskellige deployments
- Brug forskellige porte og giv adgang via; "sudo nano /etc/nginx/sites-enabled/default"
- Restart serveren og brug auto startup kommandoen forklaret længere oppe "pm2 startup"

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
 - Se oven over igen "pm2 startup".
- Ensure that you Node-process restarts after a server (Ubuntu) restart
 - <https://www.digitalocean.com/community/tutorials/how-to-use-pm2-to-setup-a-node-js-production-environment-on-an-ubuntu-vps>
- Ensure that you can run "many" node-applications on a single droplet on the same port (80)
 - Brug sub domæner.

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

- gameProject/Middleware/simpleLogger.ts eller myCors.ts
- Funktioner der har adgang til req, res og next.
- Express kan bruge router-level middleware
- MiddleWare funktioner kan:
 1. Eksekvere kode.
 2. Ændre på req og res objekter.
 3. Stoppe req-res cycle.
 4. Kalde den næste middleware funktion på stakken.

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

- Manuel test: Postman
- REST-API: gameProject/userApi.ts
- Test: gameProject/userEndpointTest.xx
- Vi gør brug af routers, som hjælper os med at holde styr på vores api stier. Her til kalder vi en middleware callback function, så vi kan manipulere med vores req og res objekter.
- Vi bruger node-fetch til at kalde vores api og lægger det fetch'ed ind i en variable i form af json. Herfra kan vi så teste på om vores variable indeholder det vi forventede.

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

- gameProject/userFacadeDBTest.ts
- gameFacadeTest.ts, der bruges chai når vi skriver expect().to.be..., som gør det mere læseligt for andre, selv ikke-programmører.

NoSQL and MongoDB

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

- En NoSQL DB indeholder ikke et schema, hvilket gør at man kan kaste alle typer af information efter den samme DB, uden at man skal opdaterer schemaet. Der er ingen relation, eller hvertfald få i en NoSQL DB, da det er semi-structured. En NoSQL DB er et dokument, som store de forskellige data. En NoSQL DB anvender collections, frem for tables.

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

- Pros:
 1. Fordi der ikke er nogen struktur i en NoSQL DB, kan man nemt gemme en masse data, som ikke nødvendigvis passer sammen.
 2. Du kan gemme alt.
 3. En klar fordel hvis du blot skal gemme informationen, og ikke hente/læse fra DB.
- Cons:
 1. Det kan være svært at lave opslag/søgninger i en NoSQL DB, da der ikke er nogen relationer eller nogen anden form for struktur
 2. Det er svært at rette/update i en NoSQL da dette skal gøres på hvert enkelt entry i Db'en, som er berørt af ændringen.

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.

- A 2dsphere index supports queries that calculate geometries on an earth-like sphere. 2dsphere index supports all MongoDB geospatial queries: queries for inclusion, intersection and proximity.

<https://docs.mongodb.com/manual/core/2dsphere/>

- TTL indexes are special single-field indexes that MongoDB can use to automatically remove documents from a collection after a certain amount of time or at a specific clock time. Data expiration is useful for certain types of information like machine generated event data, logs, and session information that only need to persist in a database for a finite amount of time.

<https://docs.mongodb.com/manual/core/index-ttl/>

- TLL - Expires after 30 seconds. Is what we specified in our program. Can be seen in the example below as "EXPIRES_AFTER"
- 2dsphere - Helps us calculate in a more precise way GEOdata. As you cannot make a sphere flat without "stretching" the map.
- gameProject/gameFacade.ts


```
await positionCollection.createIndex({ "lastUpdated": 1 }, { expireAfterSeconds: EXPIRES_AFTER });  
await positionCollection.createIndex({ location: "2dsphere" });
```

Demonstrate, using a REST-API *designed by you*, how to perform all CRUD operations on a MongoDB

- Se nederst i app.ts for egen kode. (gameProject)
- Demo brug Lars userAPI.ts. Brug postman. Eneste der mangler er PUT(U i CRUD).


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

- Se userFacadeTest.xx. Husk at indkommenter klasse(.ts)(gameProject)


 Demonstrate, using your own code-samples, decisions you have made regarding → normalization vs denormalization

- gameProject/makeTestLocationData.ts.
 - Vi denormalisere vores userName/name, for at vi ikke skal gå ud og sætte relationer op i form af id'er. Dette gør vi for at sparer på kompleksitet af koden. Vi gør dette så vi ikke skal ud og finde ud af hvilket id der tilhører hvilken user, i to forskellige tabeller, så et join statement er sparet væk.
 - Ulempen ved dette er at hvis vi skulle rette i den information der er stored på useren, så skulle vi rette i begge tabeller "manuelt" (user og position)
- Normalization er når vi joiner eller benytter en key fra en anden tabel, og derfor laver en form for "relation".
- Denormalization er når vi har redundant felter/variabler i flere tabeller.
 1. Dette kan betale sig i dag, da serverplads er utroligt billigt og man derfor godt kan "spilde" plads på at have den samme kode igen og igen.


Geo-location and Geojson

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


- Se Uge5/Week5 gameData

 Explain and demonstrate ways to create Geo-JSON test data


- makeTestLocationData.ts gameProject. Made by Lars.

 Explain the typical order of longitude and latitude used by Server-Side API's and Client-Side API's

- Server side: long/lat
- Client side: lat/long

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

- Se Uge5/Week5 app.js
- <https://gps-coordinates.org/distance-between-coordinates.php> to show it does the correct math, to show distance to user.
- Geolib: gju

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

- gameProject/gameAPI/Nearbyplayers
- I gameAPI.js findes post /nearbyPlayers, som har lon og lat som koordinater, de sendes videre til gameFacade.nearbyPlayers, som benytter MongoDB's geospatial queries og indexes.

Kan testes i postman:

POST: <http://localhost:5555/api/gameapi/nearbyplayers>

```
{
  "userName": "t1",
  "password": "secret",
  "lat": 55.77,
  "lon": 12.48,
  "distance": 10000
}
```

This will come in period-3

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

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