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, C:\Users\rasmu\Desktop\Datamatiker 2021\FullStack JavaScript\P2-Handin (period 2)\Dag 4 - Testing with Mocha, chai and Supertest\startcode-main\test\friendFacadeTest.tsfor example, Java/JAX-RS/Tomcat

Hvis man sammenligner de to med hinanden. Vil den største fordel være tid. Det er langt hurtigere at opsætte og bruge Node.js + Express, ift. Tomcat/JAX-RS/Java.

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

- Debug: Kun I konsol ikke gemt I filer.
- Appilication logging: Logging til fil I production, mulighed for at fjerne dette I debug.

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

- I starten valgte at fokuserer mere på GraphQL, herved fjernet logging. Dog kan man putte I App.ts og www.ts packages som "Simplelog" eller vores middleware logger (Morgan) som vi bruger til at implementere vores (Winston) som middleware.
- Exempel på www.ts:

```
app.get("logger")
app.log("info", `Connection to ${process.env.DB_NAME} established`);
```

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

I dag 4 som omhandler Testing with Mocha, chai and Supertest.

Link til Test Endpoint: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/test/friendFacadeTest.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.

- .env fil: Her ligger vores environment values eller “secret values”.
- I startkoden er logger sat til app objektet. Morgan er herved blevet implementeret til at bruge Winston som middleware. Dette er gjort ved at morgan er opsat til at kigge på app.
  Som man kan se er er disse sat som “Info”. Da disse logs er omhandler requests “GET”. Det er muligt at opsætte flere typer af levels af fejl/Info.
  Ved runtime ser det sådan ud:

```
21-06-2021 14:56:54:5654 : 'info' GET / 200 5.814 ms - 327

21-06-2021 14:56:54:5654 : 'info' GET /myjavascript.js 404 1.520 ms - 154

21-06-2021 14:57:07:577 : 'info' GET /api/friends/all 304 90.187 ms - -

21-06-2021 14:57:38:5738 : 'info' GET /api/friends/me 304 67.429 ms - -

21-06-2021 14:58:06:586 : 'info' GET /api/friends/me 200 66.048 ms - 187

21-06-2021 14:58:17:5817 : 'info' GET /api/friends/all 200 61.665 ms - 122
```

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 used, how authentication and authorization is handled

- **MongoDB: er opsat I filen dbconnector. Her har vi en rigtig database med vores connection string, samt en memory database vi burger til vores tests.**
- **Authentication: Dette burger vi som middleware, hvor vi har opsat det til /friends endpoint. Dvs. Alle endpoints som indeholder friends kræver authentication. Derudover**

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.

- App.ts: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%203%20-%20NoSQL%20and%20MongoDB/startcode-main/src/app.ts

- FriendRoutes: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%203%20-%20NoSQL%20and%20MongoDB/startcode-main/src/routes/friendRoutesAuth.ts

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

- Middleware: https://github.com/Erayous/Fullstack-Javascript/tree/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/src/middleware
- App.ts: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/src/app.ts

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

- **FriendsRoutes: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/src/routes/friendRoutesAuth.ts**

- **Test backend: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/test/friendBackendTest.ts**

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

Som jeg også er kommet ind på før. Vi har lave ten friendFacade. Som indeholder alle vores funktioner til at håndtere vores friends. Hertil burger vi Schema og collections fra mongodb.

- Link til friendFacade: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/src/facades/friendFacade.ts
- Link til friendFacadeTest: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/test/friendFacadeTest.ts

NoSQL and MongoDB

***Explain*, generally, what is meant by a NoSQL database.**
Forskellen på en NoSQL o gen normal SQL database er at NoSQL tiknytter sig med en key og en value. Man kan tilgå de forskellige instancer vha. af keyen. Her vil det være muligt at tilknytte forskellige values til forksellige intancer, hvilket ikke er muligt I en SQL, da du bliver nød til at definerer de forskellige kolonner.

Selve strukturen er tables(dokumenter) som bliver hashed og gemt I flere partitioner på selve serveren.

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

- SQL: Databaser er vertical skalerbare. Man kan opgraderer serveren CPU, memory mm. Selve strukturen er som vi kender det med rows og kolonner.
- NoSQL: Bruger dokumenter, hvor man kan tilføje det vil man I hvert document. De er horizontal skalerbare, da man kan oprette flere af dem, og herved klare langt mere traffic. Også kaldet "sharding".

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

- `db.collection.createIndex( { name: -1 } )`
  Denne metode laver et indeks af vores nøgle (key) I aftagende indeks, I vores navn.

Bruger man som vi har gjort I kodeeksemplerne, vil man sætte dette som

- `this.friendCollection.createIndex({ email: 1 }, { unique: true })`
  Da emailen skal være unik pr. Friend.

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.

- **TTL:** TTL indexes er MongoDB speciel indeksering, som MongoDB burger til at identificerer hvornår dokumenter skal udløbe efter tid. Dette kunnne være logs eller session information. (Ikke noget jeg har brugt I min kode).
- **2dsphere:** Er MongoDB indeks for at beregne geometri, herunder queries for inclusion, kryds og nærhed. (Ikke noget jeg har brugt I min kode).

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

- Dette har vi gjort I vores façade.
  https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/src/facades/friendFacade.ts

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

- friendBackendTest: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-

main/test/friendBackendTest.ts

- friendFacadeTest: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/test/friendFacadeTest.ts

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

- Mocha: Selve test frameworket.
- Chai: Bibliotek I stedet for nodes standard assert. (Expectation)
- Supertest: Er vores bibliotek til at lave tests op mod vores endpoints. Dette burger til at lave simple https requests.
- Nock: Dette burger vi til at tjekke eller "intercepte" vores http requests. Man laver et "Mock" af det vi får tilbage (response) da man ikke kan være sikker på at vi får det samme resutalt tilbage.

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

Her har vi brugt Nock. Som er demonstreret I mappen "playground".

- https://github.com/Erayous/Fullstack-Javascript/tree/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/playground

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.

- friendBackendTest: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/test/friendBackendTest.ts

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

- friendBackendTest: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/test/friendBackendTest.ts
- friendFacadeTest: https://github.com/Erayous/Fullstack-Javascript/blob/main/P2-Handin%20(period%202)/Dag%204%20-%20Testing%20with%20Mocha%2C%20chai%20and%20Supertest/startcode-main/test/friendFacadeTest.ts