Back-end exam reading list

by Juhani Välimäki

These materials are meant to be looked at together with the case codes, and any given demo codes. Trying to see the connection of each mentioned thing with the things in the project. Either look at them before project work or during, and also after the work while reflecting upon the project.

1. **Orientation** to how much there is going on in client – server applications (**full-stack architecture** applications) and examples of the multitude of things the front and the back must be capable of handling. Server means here the whole backend, e.g. two computers: app server + db server:

https://github.com/valju/docs_backend_design/blob/master/Responsibilities_services_or_components_ClientCarrierServer.pdf

2. Orientation continues with the three-tier architecture: client (usually web or mobile, could be desktop or console app) – application server – database server. Here mostly look were the following aspects are tackled: **user experience, performance, security**.

https://github.com/valju/docs_backend_design/blob/master/WhatToDoInFrontBackAndDB_usability_performance_security/What%20to%20check%20where%20in%20full-stack.pdf

This explains a bit of the access control jungle of full-stack apps:

https://github.com/valju/docs_backend_design/blob/master/WhatToDoInFrontBackAndDB_usability_performance_security/naturalpeople_webuserlogins_approles_dbusers_dbroles_owndbitems.pdf

3. An important concrete topic to introduce or recap. The **asynchronous nature** of modern code and systems. E.g. the Nodejs backend. (Or AJAX in frontend)

https://github.com/valju/docs_backend_design/blob/master/AsynchronousOperations/AsynchronousOperations_AnExampleOfDoubleAsynchronous_ExecutedInThreeDifferentMoments.pdf

4. Important points about test data. It's important e.g. for the programmer or a UX evaluator.

https://github.com/valju/docs_database_design/blob/master/TestData_Creation_ImportantToKnow.pdf

5. The database operation return values and error code interpretation. How the backend and RDB database server "communicate" with each other. This code is not perfect, but has some good components to take and refine. Look e.g. at single Category creation (POST), deletion (DELETE), update (PUT), single select (GET), get all (GET).

https://github.com/valju/idea-case-backend/blob/master/src/routes/api/category.js

https://github.com/valju/idea-case-backend/blob/master/BackendDemoProject_ForExam.pdf

6. An example of some of the steps that might happen while *defining*, *designing and building* backends. A bit similar to our case, but remember that there are no two companies, often no two projects even that would be the same. This is again tool for organizing the thoughts, and to know how to gather some of the needed skills in a step list.

https://github.com/valju/docs_backend_design/blob/master/BackendCreationStepsExampleBitSimilarToOurCase.pdf

7. Basic knowledge of **Http Methods**, **Http Status codes**, and **TCP ports** (No doc links, just these)
   - Http Methods, 4 we used: GET, POST, PUT, DELETE (+PATCH)
   - Http Status codes, in general, 2XX ok , 3XX redirect, 4XX client/request originated prob, 5XX server-side related prob.
     - 200 ok,
     - 400 generic request-based error, when no other more suitable exists or no specific wanted. 401/403 Unauthorized/Forbidden, 404 resource not found, 409 conflict (primary or alternate key=duplicate entry), When need the other ones, check from lists.
     - 500 generic server side prob. When need the other ones, check from lists.
   - TCP (no UDP this time) ports we use: about 5-6, (:22 SSH, :80 HTTP, :3000 Node.js default port, :3306 MariaDB/MySQL, :443 HTTP**S**, :8080 HTTP Proxy)
   - We used 8787, 8686 instead of 3000 to not to have several node servers running is same port, We used 3308 so that the tunnel would not hide your own other local Mariadb/mysql
   - Because: only one server/service/thread can listen to a port at a time. Unless you use e.g. nginx to relay requests coming in through one external port to go to different internal ports, based on the URL

8. **About web services**: (1970s EDI),(SOA), (SOAP, XML, WSDL), ESB, REST API, **REST API principles**, microservices, **GraphQL**, serverless computing/ serverless functions)
   https://github.com/valju/docs_backend_design/blob/master/REST_Api_services/WhatAreWebServices_SOA_SOAP_RESTful.pdf
   (Not in this material, but could be: web sockets (older concept), webhooks, server sent events…)

9. Architecture topic intro:
   https://github.com/haagahelia/swd4tn023/blob/master/06_ohjelmistoarkkitehtuurit_ja_patternit/SoftwareArchitecturesAndPatterns.pdf

10. *This task drawn on a A3 paper, like practicing for job interviews: "Draw and explain the archictecture of that school project you were in".*
    About third of the exam points: FULL-STACK ARCHITECTURE DRAWING AND EXPLANATION
    BASED ON THE GIVEN PICTURE: *) and SAMPLE EXPLANATIONS: (Make your own better ones)

    *) Think about Front-end max this much before Backend exam/Full-stack architecture pic:
    - (Axios used to run) AJAX to backend REST API. (Http Requests (+JSON in request body, or GET params in URL?) to carry data/parameters)
    - React app (based on the create-react-app template) with React-router SPA (Single-page application) routing

11. CORS https://en.wikipedia.org/wiki/Cross-origin_resource_sharing
    Becomes an issue to handle in most of our projects. Why? Because originally the web browser will get the first page of the SPA app from the web server (Origin). Then the page will AJAX-call the REST API services from the backend (=> content from a different 'Origin')
    (Or if you would have your react front and node backend on localhost, one would be localhost:8686 and other localhost:8787, two different processes, different 'origins' still.)

12. Foreign policy selection for business cases (Foreign policies learned on previous courses)
13. ((Practical sample case tool knowledge: https://github.com/haagahelia/linux-servers-etc/))

(Always read the latest version of any docs: git pull, git pull, git pull)

NOT THIS TIME:

14. Message queues as architectural model (to following exams, not yet 29.9.2022)
15. ((((Full-stack open course reading list. Some of the bright blue marked might lead to Backend exam questions.