Period 3 GraphQL

Lavet af Christian Madsen, Frederik Hurup & Jacob Simonsen

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

Explain shortly about GraphQL, its purpose and some of its use cases

GraphQL er et syntax sprog hvor man kan spørge efter lige præcis det data man gerne vil have, hvilket gør at man skal sende færre og mindre tunge requests for at få den ønskede data. Det gør det også muligt at hente data fra flere forskellige kilder på en nem måde.
Mobile enheder og IOTs får det meste ud af en måske dårlig forbindelse ved at mindre datatrafik.

Explain some of the Server Architectures that can be implemented with a GraphQL backend

Der er to måder at implementere GraphQL på backend:
Det kan lægges som en "skal" uden om et eksisterende REST-api, hvor GraphQL kan bruges til indsamle data fra de forskellige endpoints.
Det kan også bruges til at kommunikere direkte med serveren, hvor GraphQL så er et enkelt "REST-endpoint", som kan hente al den data serveren tilbyder.

What is meant by the terms over- and under-fetching in GraphQL, compared to REST

Det tillader udvikleren at hente specifikt data så det ikke bliver så data tungt for api'et. Over-fetching, er hvor man er nødsaget til at hente mere data end hvad der reelt er brug for. Under-fetching, er hvor man ikke får tilstrækkelig data og derfor nødsaget til at lave flere requests, fra flere endpoints.

Explain shortly about GraphQL's type system and some of the benefits we get from this

Man kommunikere med GraphQL igennem Query og mutation typer. Man definere typerne i schema'erne, om det er object eller basic types. En af de gode ting er at man selv kan sætte sine rammer op for det man skal have ind eller ud. Ud for hvert argument kan man definere om de er optional eller not nullable.

Explain shortly about GraphQL Schema Definition Language, and provide examples of schemas you have defined.

Det er en måde at lave sit eget regelsæt med navne på de kald der kan laves, og hvad man kan få ind som input samt hvad få ud som output.

Se eksempel på schema fra startcode backend: src/graphql/schema.ts

Provide examples demonstrating data fetching with GraphQL. You should provide examples both running in a Sandbox/playground and examples executed in an Apollo Client

se eksempel på src/components/FindFriend.tsx fra startcode frontend

Provide a number of examples demonstrating; creating, updating and deleting with Mutations. You should provide examples both running in a Sandbox/playground and examples executed in an Apollo Client.

Se eksempel på src/components/AddFriend.tsx for både creating og updating.
Se eksempel på src/components/DeleteFriend.tsx for deleting

Explain the Concept of a Resolver function, and provide a number of simple examples of resolvers you have implemented in a GraphQL Server.

En resolver function udfører querys og mutations der er defineret i schema'et ved at bruge de samme navne.

Se eksempel på resolver fra startcode backend: src/graphql/resolvers.ts

Explain the benefits we get from using a library like Apollo-client, compared to using the plain fetch-API

Apollo-client gør det muligt at bruge useQuery som er et REACT hook, som bruger hooks api til at dele GraphQL data med UI.

In an Apollo-based React Component, demonstrate how to perform GraphQL Queries, including:

- **Explain the purpose of ApolloClient and the ApolloProvider component**

ApolloProvider svarer til React's context.provider. ApolloProvider wrapper React App'en og placere klienten på contexten der gør der muligt at tilgå alle steder fra i komponent træet.

- **Explain the purpose of the gql-function (imported from graphql-tag)**

Den gør det muligt at definere graphQL's forespørgsler og execute requestene via useQuery.

- **Explain Custom Hooks used by your Client Code**

Custom hooks er functions som kan kalde andre hooks, såsom useState, hvilket client koden bruger meget. Det er f.eks. brugt i AddFriend.tsx til at holde den friend der skal tilføjes.

- **Explain and demonstrate the caching features built into Apollo Client**

Det er en form for at en lille database i clienten, som kan huske data, såsom allFriends, og derfor behøver der ikke komme kald til serveren hver gang brugeren forlader allFriends siden og kommer tilbage igen. Det bliver lagt i cachen. Den kan manipuleres, som vi gør når der bliver tilføjet en ny friend. I stedet for at skulle hente alle brugere fra DB hvor den nye friend ligger, bliver den nye friend tilføjet til cachen og dermed vist i allFriends.

In an Apollo-based React Component, demonstrate how to perform GraphQL Mutations?

se src/components/AddFriend.tsx i frontend koden.

Demonstrate and highlight important parts of a "complete" GraphQL-app using Express and MongoDB on the server-side, and Apollo-Client on the client.

Opsætning af express via middleware i "src/app.ts". Etablering af forbindelse til mongoDB på applikationsniveau i "src/bin/www.ts". ApolloProvider rundt om REACT componenterne(se tidligere svar).