PG6300 Webutvikling og API-design: Mappeelement 2

Det er en selvfølge at en organisasjon har en egen dynamisk nettside i 2016. Du skal lage en applikasjon bestående av server og webklient for selskapet PP Records BRL ASA Inc., som lister opp alle album de har tilgjengelig. En liste over album finnes i fila albums.txt. I tillegg skal du kort besvare noen spørsmål.

Oppgaven bygger videre på mappeelement 1. Du står fritt til å ta utgangspunkt i din egen besvarelse, eller løsningsforslaget som finnes på https://github.com/theneva/pg6300-15-a02-starting-point.

Spørsmål

Besvarelsen på denne oppgaven bør totalt være på 250–300 ord.

1. Hva er noen fordeler og ulemper ved å bruke Redux i stedet for lokal state i React-komponenter?
2. Hva er viktige forskjeller på en dokumentdatabase (som MongoDB) og en relasjonsdatabase (som MySQL)?
3. Forklar kort oppgavene til følgende komponenter med dine egne ord:
    1. Mongoose
    2. bcrypt
    3. JSON Web Token (JWT)

Features

Webklienten skal bestå av én side, der innhold vises frem avhengig av om brukeren er innlogget eller ikke. Komponenter skal gjenbrukes der det passer seg. Det skal være mulig å:

- Dersom man ikke er logget inn
    - Se et skjema for å registrere seg som ny bruker eller logge inn
    - Se de 10 nyeste albumene (artist, tittel, og årstall) som er publisert, og hvilken bruker som eier (opprettet) hvert album
- Dersom man er logget inn
    - Se hvilken bruker man er logget inn som
    - Logge ut ved å trykke på en knapp

  - Se en liste over (bare) sine egne albumer
  - Legge til et nytt album
  - Slette et album (for eksempel ved å trykke på en knapp ved siden av hvert album)
- Uavhengig av innloggingsstatus
  - Sortere lista over albumer på enten artistnavn eller årstall
  - Filtrere ("søke i") lista ved hjelp av et input-felt. Lista skal oppdateres ved hver endring i input-feltet. Det trengs *ikke* å gjøres spørringer til serveren for å oppnå dette.

Opprettelse og sletting av albumer skal reflekteres i databasen.

Tekniske krav

- Webklienten skal benytte React for å generere all HTML mellom `<body>` og `</body>`, unntatt
  - én div: `<div id="container"></div>`, og
  - referanse til bundle-en som bygges av Webpack: `<script src="..."></script>`.
- React-klienten skal kommunisere med en Node-server ved hjelp av Fetch-API-et. Serveren skal benytte modulen Express og servere dataen som JSON, med unntak av JSON Web Token-et som kan serveres som ren tekst. Tips: husk CORS (http://npmjs.com/package/cors). Grunnleggende REST-arkitektur (level 2, altså identifiers og verb) skal følges.
- Webklienten skal bygges med Webpack. Hot reloading er *ikke* et krav, men kan være til hjelp.
- Album og brukere skal ligge i en MongoDB-database.
- Passord skal hashes med bcrypt før de lagres i databasen. bcrypt skal også benyttes for å verifisere passordene ved innlogging.
- Innlogging *skal* skje ved hjelp av et signert JSON Web Token (JWT) og npm-pakka `jwt-simple`.
- Alle tredjepartsbiblioteker (for eksempel Express og Webpack) skal deklareres på passende vis i fila `package.json`. Det skal være mulig å klargjøre og starte applikasjonen med følgende kommandoer i prosjektets rotmappe:
  1. `npm install`
  2. `npm run build`
  3. `npm start`
- Kun kildekode skal leveres som ett zip-arkiv. Dette kan for eksempel løses ved å bruke `git archive` (https://git-scm.com/docs/git-archive) forutsatt korrekt bruk av `.gitignore`-fil.

Øvrig informasjon

- Du skal lage applikasjonen på egen hånd. Det er lov til å ta inspirasjon fra internett og andre ressurser, men ikke kopier kode eller tekst uten å referere. Den største delen av arbeidet skal være ditt eget (unntatt tredjepartsbiblioteker).
- Første versjon av besvarelsen leveres på It's Learning innen 3. april 2016 klokka 23:59. Frist for oppdatering annonseres senere.
- Eventuelle spørsmål rettes per epost til martin@westerdals.no.