Projekt 2 - Quiz app

2. Semester

Anders Ravn Adriansen

16/03/2023

GitHub: https://github.com/andersravn32/quiz-app

Environment variables: https://pastebin.com/ZLBZNm1G

Indhold

Indledning

Hvad angår webudvikling som helhed kan man ikke komme uden om datasikkerhed eller generel datastruktur, da begge spiller en enorm rolle for applikationsudvikling og generel sikkerhed.
Datasikkerhed og dataintegration er derfor to fag som spiller rigtig godt sammen da man som udvikler både skal være i stand til at strukturere sine data, og sikre dem på bedst mulige vis. Med denne filosofi i tankerne blev vi derfor stillet til opgave at udvikle en web app som besøgende brugere kan benytte til at tage "quizzes". Lige præcis den her opgave har stor relevans for den tværfaglige kombination af datasikkerhed og dataintegration, da vi som udviklere er nødt til at udvikle en datastruktur som er mere kompleks end man lige umiddelbart ser ved første øjekast. Derudover skal den selv samme datastruktur også sikres på bedst mulig vis efter moderne standarder. Med dette i tankerne kan der stilles en problemformulering.

Problemformulering

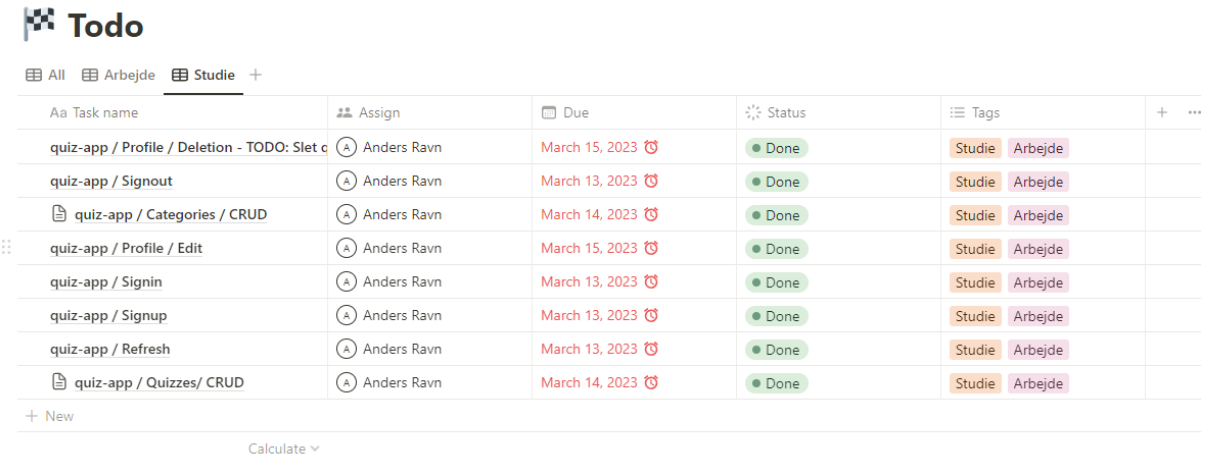
Hvordan kan der på sikker vis udvikles en quiz app med dertilhørende brugergrænseflade og backend?

Metodeovervejelser

Før det generelle udviklingsarbejde kunne påbegynde skulle jeg beslutte mig for hvordan arbejdet generelt set skulle forløbe.
Jeg kunne forud for udviklingen allerede identificere en klar struktur i projektet, samt forstå de krav der blev stillet. Det var derfor åbenlyst at benytte mig af vandfaldsmetoden**,** da den understøtter den konstante fremgang jeg gerne vil have, da jeg anser projektet som værende "rutine" da kravspecifikationerne ikke går meget ud over hvad jeg har prøvet før. Derudover så er vandfaldsmetoden også rigtig god til at løse opgaver såfremt der er et tidspres, da den har større fokus på det generelle resultat, frem for andre arbejdsmetoder som f.eks. den agile arbejdsmetode, da jeg heller ikke anså det som en nødvendighed at arbejde iterativt i dette projekt.
Forud for at udviklingen kunne begynde skulle jeg også beslutte mig for en række teknologier som jeg kan benytte mig af til at løse opgaven tilfredsstillende. Disse teknologier er delt i to kategorier, frontend- og backend udvikling.
Til backend udvikling benytter jeg mig af Express.js som generel server applikation. Dette resultere i en forudsigelig struktur hvor meget er givet på forhånd rent strukturmæssigt. Derudover benytter jeg mig af et ganske simpelt login flow som benytter JSON web tokens til håndtering af sessions.
Da det blev givet som krav at backend applikation skulle have såkaldte "API endpoints" benytter jeg også cors til at seperere den trafik som backenden nu engang skulle modtage således at mine systemer er delt korrekt op efter hvilken kilde anmodninger er sendt fra. Til slut benytter jeg almen MongoDB som database, og får hostet min database gennem MongoDB's egen tjeneste, Atlas.
Til frontend udvikling benyttede jeg mig af en kombination af Vue.js og Pico.css. Vue.js muliggør at lave layouts og responsive brugergrænseflader på en hastighed som simpelthen ikke er muligt med almen JavaScript kode, hvilket jeg vægtede virkelig højt i dette projekt. Derudover så benytter jeg også Pico.css som er en forholdsvis ny teknologi som ligesom Vue.js også giver et massivt hastigheds boost på hvor hurtigt det er muligt at udarbejde layouts og komponenter til brug i din app.

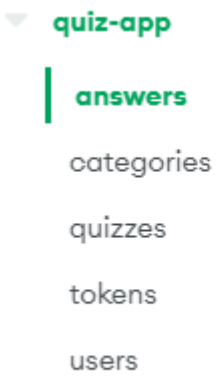
Research

Forud for udviklingens begyndelse skull jeg fastlægge hvilke delopgaver selve projektet bestod af. Dette gjorde jeg gennem et eksternt projektstyringsværktøj ved navn Notion. Jeg delte de forskellige aspekter af projektet op i delopgaver som alle havde til formål at bidrage til med små bidder til det store billede i sin helhed.



Derefter skulle jeg have placeret forskellige deadlines på de individuelle opgaver. Dette blev gjort ud fra en generel vurdering af hvilke aspekter af opgaven der vægter mest, samt hvilke ting er mest kritisk for de resterende punkters success. Alt det ledte mig til en færdig liste over delopgaver som alle skulle løses før at projektet kunne anses som værende færdigt.

Udover generel planlægning skulle jeg også beslutte mig for en generel datastruktur som jeg vil kunne benytte til netop dette projekt. MongoDB tillader at jeg kan arbejde forholdsvis flydende, hvilket jeg anså som værende et stort plus. Det viste sig dog at funktioner som SQL tilbyder f.eks. JOINS og relationer ville have været ganske nyttige i dette projekt, men da MongoDB ikke tilbyder det umiddelbart, måtte jeg arbejde udenom dette gennem renskrevet server logik i stedet.



Jeg kom frem til en generel struktur som bygger omkring fem forskellige collections.

De to collections, users og tokens, benyttes udelukkende til authentication og authorization, da det er her både brugerdata og tokens til session styring osv ligger gemt.

Det er derfor de resterende tre collections som er mest interessante at se på.

- Categories, indeholder som navnet antyder alle forskellige typer af kategorier af quizzes som bliver udbudt. Disse kategorier bliver så senere læst og benyttet til videre oprettelse af quizzer.
- Quizzes, indeholder alt data vedrørende diverse quizzer som bliver udbudt. Dette inkluderer også hvilken kategori quizzen tilhører, samt hvem der har lavet og de tilhørende spørgsmål, svarmuligheder på disse spørgsmål, samt de korrekte svar. Questions arrayet i de individuelle quiz objekter følger også en specifik struktur som har til formål at gøre det nemmer at sammenligne svar, som ligger i en seperat collection, på et senere tidspunkt.

```
{
  _id: null,
  name: null,
  category: // Reference to category id,
  creator: // Reference to user id
  questions: [
    {
      question: null,
      options: [
        {
          option: null,
          correct: false
        },
        {
          option: null,
          correct: false
        },
        {
          option: null,
          correct: false
        },
      ]
    }
  ],
  public: false
}
```

- Answers indeholder alle former for svardata som et direkte modsvar på de individuelle questions arrays som brugerne har besvaret.

Det skal dog siges at quizzes og answers ikke har noget direkte relation som ellers ville være muligt ved brug af SQL, og dermed skal matematikken bag procentvis score og generelt data alt sammen beregnes, enten ved anmodning til serveren, eller direkte på client siden af applikationen, hvilket måske ikke vil være det mest optimale ved store mængder af data.

Konstruktion

Under udviklingen gjorde jeg mig også mange tanker i forhold til at arbejde så organiseret og struktureret som overhovedet muligt. Dette kommer til udtryk gennem noget så simpelt som mappe struktur i mit projekt, hvor ating er klart opdelt i client/server og der bliver fulgte en konsekvent navngivningskonvention hele vejen igennem projektet, både på client og server. Udover blot struktur i selve projektet har jeg også arbejdet meget med at have god struktur hvad versionsstyring angår. Her benytter jeg GitHub med klart navngivne commits og klare commit beskeder som gør det muligt at se lige præcis hvad de enkelte commits indeholder. Således er det muligt at identificere lige præcis hvor noget er gået galt såfremt jeg skulle møde udfordringer undervejs.
Selve udviklingsprocessen foregik forholdsvis smertefrit, selv på trods af et stort tidspres. Både client og server applikationerne blev udviklet jævnfør den plan jeg lagde mig i starten af forløbet. Jeg var dog nødt til at nedskalere min oprindelige ide for API delen af applikationen markant grundet det førnævnte tidspres der var til stedet gennem hele processen. Dette resulterede i at jeg ikke nåede at udvikle et fuldt REST api til at tilgå alle informationer i databasen som jeg egentlig havde ønsket. Jeg nåede kun at lave funktionen til at generere nye API nøgler og en enkelt rute og middleware til validering af selvsamme token.

Evaluering

Generelt set har hele udviklingsforløbet fungeret godt, jeg har følt at jeg har nået de fleste af de ting jeg gerne ville have nået, samt at jeg er tilfreds med det endelige resultat jeg står tilbage med.
Modsat tidligere projekter arbejdede jeg denne gang uden en gruppe, det havde til effekt at arbejdspresset på mig selv har været stort, men ikke uoverskueligt. Jeg har hele tiden haft fokus på hvordan jeg selv ville have tingene, hvilket også medvirkede til at jeg kunne arbejde uafbrudt og mere koncentreret end jeg har haft mulighed for førhen.
Naturligvis mangler selve gruppe aspektet af arbejdet, og det havde været rart at have nogen at spare med, ligesom jeg har haft i tidligere projekter, dog skal det siges at jeg har nydt udelukkende at stå til ansvar for mig selv og ingen andre. Det har medvirket til

at jeg har kunne udvikle en kompleks applikationsstruktur på rekordtid, da jeg ikke har skullet forholde mig til forhold end dem jeg selv var bekendt med på forhånd. Alt dette har resulteret i at jeg står tilbage med client og server applikationer jeg føler jeg er tilfreds med, og som opfylder langt de fleste af de givne kravspecifikationer.

Konklusion

Jeg har i dette projekt prøvet kræfter med at arbejde på egen hånd i langt højere grad end jeg tidligere har gjort i projektarbejde, hvilket har medvirket til at jeg har fået et langt større udbytte af projektet end jeg ellers tidligere har fået af andre projekter.
Generelt set er jeg meget tilfreds med de applikationer jeg har fået udviklet, både generelt set fra et teknisk standpunkt, især med det generelle tidspres i mente. Jeg har fået afprøvet nogle nye koncepter når det kommer til udvikling af API'er og prøvet kræfter med endnu engang at have det helt store overblik over et projekt, småt som det nu engang er.
Derudover har jeg også prøvet kræfter med at arbejde med et projekt som på givent vis er "rutine" arbejde, når projektet deles ud i små bidder.

Referencer

auth0.com (n.d.) JWT.IO [online] available from <http://jwt.io/> [15 March 2023]

Evan, Y. (n.d.) Vue.Js - The Progressive JavaScript Framework | Vue.Js [online] available from <https://vuejs.org/> [15 March 2023]

Lucas, L. (n.d.) Pico.Css • Minimal CSS Framework for Semantic HTML [online] available from <https://picocss.com> [15 March 2023]

MongoDB, Inc. (n.d.) MongoDB: The Developer Data Platform [online] available from <https://www.mongodb.com> [15 March 2023]

Mozilla (2023) Cross-Origin Resource Sharing (CORS) - HTTP | MDN [online] available from <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS> [15 March 2023]

npm (2018) Cors [online] available from <https://www.npmjs.com/package/cors> [15 March 2023]

OpenJS Foundation (n.d.) Express - Node.Js Web Application Framework [online] available from <https://expressjs.com/> [15 March 2023]

Visure Solutions (n.d.) Softwarekrav: Agile vs Waterfall Methodology - Visure Solutions [online] available from <https://visuresolutions.com/da/requirements-management-traceability-guide/software-requirements-agile-vs-waterfall/> [15 March 2023]