Projekt 2 – Quiz App

2. Semester

Amir Skoric

16.3.2023

GITHUB LINK: https://github.com/amir-skoric/quiz-app

Contents

Indledning

Datasikkerhed og dataintegration er to begreber der går hånd i hånd. Uden datasikkerhed burde det næsten være ulovligt at gemme sårbare data på internettet. Hvordan forhindrer vi hackere i at få adgang til vores data? Det gør vi nemlig ved at have nogle security measures.

Med denne viden har jeg udarbejdet en webapplikation med en NoSQL-database, der kan være fleksibel og skalerbar. Det jeg har gjort i ”sambarbejde” med Express JS, Express Handlebars (frontend), MongoDB (Mongoose biblioteket), og Express Sessions.

Derudover har jeg også brugt bcrypt for at hashe og salte mine gemte passwords med i databasen.

Pga. tidspresset har jeg også brugt Pico CSS som CSS library. Det gjorde det hele meget hurtigere. Jeg vil med de værktøjer forsøge at opbygge en webapplikation der både er brugervenlig, og funktionel.

Problemformulering

Hvordan kan jeg udvikle en quiz-app med en ”reactive” brugergrænseflade og med en NoSQL databasen der sikrer mig en sikker datastruktur og en brugervenlig brugergrænseflade?

Metodeovervejelser

Jeg valgte at bruge den agile arbejdsmetode, hvor jeg tog problemerne an som de kom, og rettede dem på stedet. Det var væsentligt vigtigt for mig for at få opnået de krav jeg blev stillet.

MongoDB blev brugt som database baseret på hvordan databasestrukturen er. Databasestrukturen er meget mere fleksibel og ”fri”, fremfor hvis jeg havde brugt for eksempel MySQL. MongoDB gjorde det muligt for mig at ændre ”skemaerne” for databasen løbende, hvilket også passer meget godt til den agile arbejdsmetode jeg havde valgt mig at tage udgangspunkt i.

Jeg anvendte Express Handlebars til udviklingen af min frontend. Ved brug af de indbyggede ”helpers” kunne jeg nemt og hurtigt få skabt en reaktiv og dynamisk hjemmeside. Derudover

brugte jeg som nævnt tidligere, Pico CSS til stylingen (selve udseendet/temaet) af webapplikationen uden ekstra arbejde.

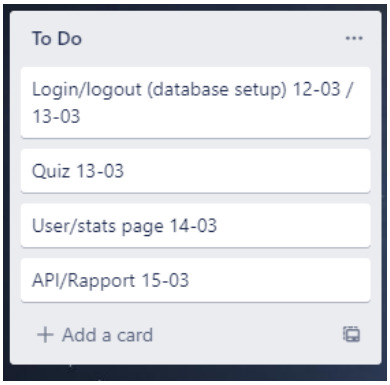
Express JS brugte jeg som backend, da det gjorde det meget nemmere for mig at udvikle en sikker backend, fremfor hvis jeg havde lavet det med en almindelig http server. Express gjorde nemlig meget af det hårde arbejde for en. Det gav mig også mulighed for at bruge hashing og saltings værktøjet ”bcrypt”, på en nem og effektiv måde. Authentication delen af backenden blev styret af express-sessions, hvilket også den del af arbejde en del nemmere.

Helt konkret havde valget af libraries/værktøjerne en stor betydningen for det endelige resultat, og dermed hjalp den mig også med at svare på problemformuleringen.

Research

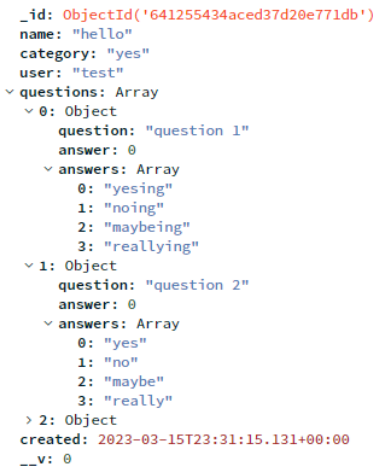
Før jeg overhovedet kunne gå i gang med at bygge min webapplikation så skulle jeg beslutte mig for hvilke værktøjer/teknologier jeg skulle gøre brug af. Jeg endte ud med at bestemme mig for at jeg helt klart skulle have en frontend, og en backend, og hvilke passende teknologier jeg kunne bruge til det.

Tidsplan



Før jeg startede på projektet, lavede jeg en lille tidsplan på Trello på hvordan mit projektforløb nogenlunde skulle foregå. Det kommer til at hjælpe mig med at finde ud af hvor lang tid jeg skal bruge på emnerne.

Quiz datastruktur



Ved at ”forudbestemme” min datastruktur som ”skema”, havde jeg det helt klart nemmere at fuldføre arbejdet end at hvis jeg ikke havde nogen form for plan.

Frontend

Som tidligere nævnt brugte jeg Express Handlebars som mit foretrækkende frontend JavaScript framework. Express Handlebars gjorde det meget nemmere for mig at få udviklet en dynamisk og reaktiv brugergrænseflade til min webapplikation. Fremfor at jeg skal sidde med rå JavaScript (uden nogen frameworks), så kan jeg nemt og hurtigt tilføje variabler og elementer fra min backend, ind til min frontend.

Jeg valgte at bruge Express Handlebars, da det lød som noget der ville spille rigtigt godt sammen med Express. Derudover kan man med Express Handlebars også lave ”moduler” (partials), som man kan smide ind i alle sine dokumenter (f.eks. en header). Det gjorde udviklingsprocessen med overskuelig, og det gjorde det også bare meget nemmere at holde styr på tingene, og undgå ekstra clutter i sine markup filer. Express Handlebars er egentlig bare en ”forlængelse” af HTML, og derfor var det nemt og hurtigt at komme ind i hvordan det hele virkede.

For at style det hele benyttede jeg Pico CSS, CSS frameworket. Jeg havde læst om det, og så at det ikke krævede nogen form for setup, det var nærmest plug-n-play.

Det var mine foreløbige overvejelser. Det skulle nemlig være nemt og hurtigt, pga. tidspresset.

Backend

Til udvikling af backenden havde jeg valgt at benytte Express som server framework. Express gjorde det helt klart nemmere at få sendt data fra backenden til frontend, og med Express Handlebars ovenikøbet, fungerede det helt fint. Express gav mig muligheden for at lave en REST API, uden en masse bekymringer. REST API giver brugeren mulighed for at tilgå quiz data i form af et http request, hvor svaret så er i JSON-formattet. Ved brug af Express fik jeg også gjort brug af Express Sessions. Express Sessions bruger jeg til at sikre mig at det kun er selve brugeren der har adgang til sine egne kontooplysninger. Man kan ikke tilgå hjemmesidens andre undersider før man er registreret og logget ind.

På baggrund af Express som server framework, valgte jeg at gøre brug af MongoDB. MongoDB fungerer med en NoSQL databasestruktur, som er simpel og skalerbar, fremfor en almindelig SQL-database, som kan skabe en masse problemer (pga. kompleksiteten) med den agile arbejdsmetode jeg brugte. Jeg håndtede dataen fra databasen med biblioteket Mongoose. Da Mongoose benytter sig af ”skemaer” (en form for blueprint), kan jeg nemt være sikker på at det jeg sender ind i databasen, kommer ind som det skal. Derudover bruger jeg også Morgan til overvågning af http requests (delay).

Brugerne på hjemmesiden har mulighed for at oprette, rediger og slette deres konti. Derudover kan de også få oprettet en quiz inde på siden, som de så kan tilgå hvor som helt, når som helst.

For at sikre brugerdataen i databasen, har jeg brugt bcrypt-biblioteket der både hasher og salter brugeren kodeord.

Analyse

Da min webapplikation er en quiz-app, skal brugeren selvfølgelig kunne oprette og tage quizzer. Gennem webapplikationen ønsker jeg at jeg kan tilbyde brugerne en god og brugervenlig brugergrænseflade, der ikke skal sætte en stopper for hvad de laver.

For at kunne opnå dette mål satte jeg mig ind i hvordan jeg kunne gøre brug af mine valgte teknologier, og hvordan jeg skulle få dem til at spille sammen. Det kunne jeg nemlig med en masse ”trial and error”. For at løst min problemformulering skulle jeg igennem en masse læring, og bevise at jeg har faktisk har gjort brug af det research og metoder jeg har sat mig ud for at bruge.

Konstruktion/Evaluering

Under udviklingen af webapplikationen skete det hele lidt spontant. Jeg har dog forsøgt i sidste ende at lave et projekt der er organiseret og opdelt i forskellige komponenter. Grundet tidspres er jeg dog ikke super tilfreds med det endelige resultat, og har mange gange igennem forløbet tænkt over hvordan har kunne gjort det anderledes. Derudover har mit brug af GitHub som versionsstyring heller ikke været særligt imponerende. Jeg sidder og koder og så glemmer jeg bare at committe.

Backenden kunne jeg helt klart have lavet på en bedre måde. Hvis jeg brugte et ordentligt framework som ”Vue.js”, så ved jeg at jeg ikke skulle blive trukket ned at HTML manglende put/delete metoder. Jeg blev nemlig nødt til at POST’e alle mine delete og put requests på en ikke så smart måde. Mine routes kunne have virket bedre. I stedet for ”form method”, skulle jeg have brugt fetch requests. Der er så mange ting der ikke gik som planlagt, men det hele er en læringsproces, og det er helt klart noget jeg vil tage med mig.

Quiz ”tilføjelses”-funktionen er også noget der blev lavet på meget kort tid, da Express Handlebars forhindrede mig i at komme videre i mit projekt flere timer ad gangen.

Jeg følte helt klart at jeg ikke har nogen konkret plan, og det vil helt klart tage med mig i imødekommende projekter.

Jeg føler dog også at jeg har lært en hel masse om hvordan man kan få backenden og frontenden til at spille sammen, men jeg manglede bare den plan og struktur i projektet, som jeg ledte efter.

Udover det, nåede jeg desværre ikke at lave en ”stats” side grundet tidspres.

Konklusion

Alt i alt får jeg prøvet noget nyt og arbejdet på egen hånd. Det er helt klart noget jeg vil gøre igen, da jeg faktisk endelig føler at jeg har fået lært noget. Jeg er nogenlunde tilfreds med det endelige produkt (den tekniske del), men ville dog stadigvæk ønske at jeg bare havde sat en bedre plan op.

Tidsplanen/fordelingen af opgaverne fungerede fint, indtil jeg stødte ind i nogle problemer der fik udskudt nogle af delene. Egentlig burde jeg have fulgt ”vandfaldsmetoden”, men endte så med at bruge den agile metode, da tingene ikke gik planlagt.

Det virker, men det er ikke optimalt, og næste gang vil jeg tage det med mig.

Jeg får opfyldt de fleste af kravene fra projektbeskrivelsen, og det må jeg acceptere er godt nok.
Jeg ved hvor jeg har fejlet, men grundet tidspres, har jeg ikke kunne have nået at rette op på det.

Referencer

Bcrypt (2022) available from https://www.npmjs.com/package/bcrypt [16 March 2023]

Express-Handlebars (2023) available from https://www.npmjs.com/package/express-handlebars [16 March 2023]

Express-Session (2022) available from https://www.npmjs.com/package/express-session [16 March 2023]

Handlebars (n.d.) available from https://handlebarsjs.com/ [16 March 2023]

Mongoose ODM v7.0.2 (n.d.) available from https://mongoosejs.com/ [16 March 2023]

Morgan (2020) available from https://www.npmjs.com/package/morgan [16 March 2023]