Library project
PBWEB Marts 2021

Nadia Skau
Morten Højrup Kristensen
Morten Due

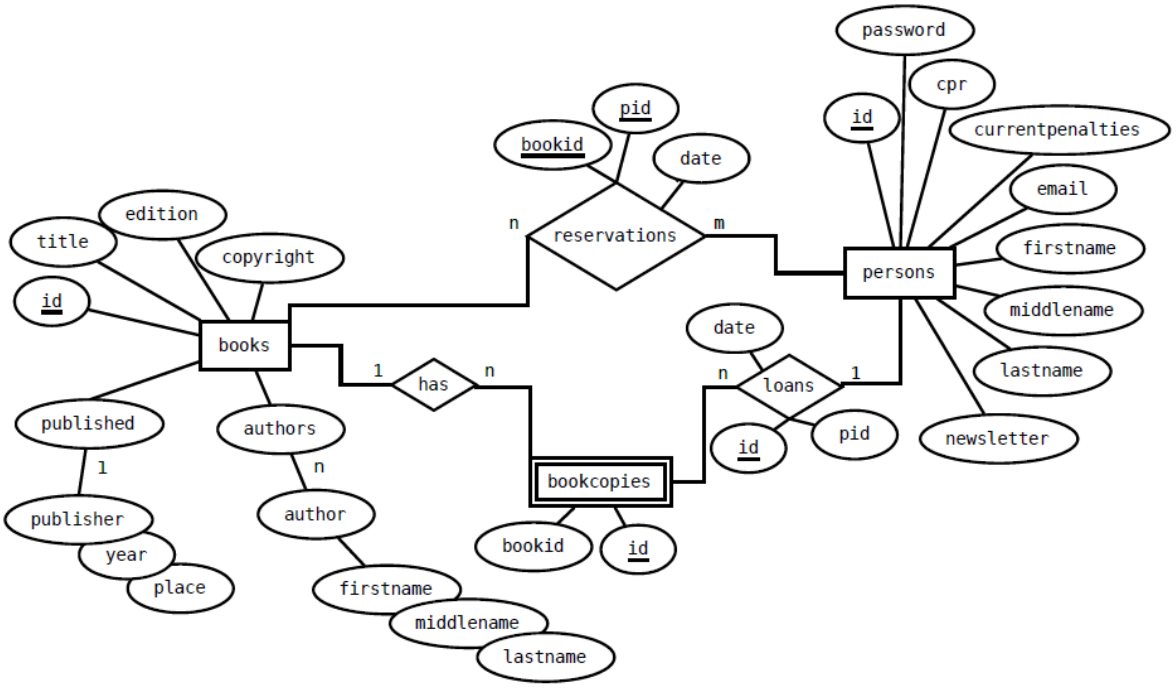
Indholdsfortegnelse

Indledning

Vi har i undervisningen gennemgået Node.js og brugen af Express og template-engines samt MongoDB og Mongoose-modulet til at håndtere data fra MongoDB.
I dette projekt skal vi bygge en webapplikation, der er et bibliotek (af bøger). Hertil har vi en Mongo database at gå ud fra med fem collections samt et repository med en Express applikation.
Applikationen skal kunne håndtere følgende:

- udlån
- reservation
- aflevering
- bibliotek af bøger (opbevaring af information)
- opbevaring af brugerinformation
- log-ind

Herunder ses ER-modellen for den givne database:



Problemformulering

Vi vil i denne opgave udvikle en web-applikation, der agere som et fuld funktionelt bibliotek. Brugere skal have mulighed for at oprette sig på siden, logge ind, foretage reservationer og oprette lån af bøger.
Desuden skal brugeren få fremvist en liste over bøger, og kunne skelne mellem udlånte og ledige bøger med mulighed for at reservere udlånte bøger.
Dertil vil vi i denne opgave, besvare nedenstående spørgsmål.

Hvordan får vi fremvist bøgerne på en overskuelig måde?
Hvordan skal brugeren kunne fremsøge bøger?
Hvordan opbevarer vi bruger informationen sikkert?
Hvordan håndterer vi følgende:

- Opret ny bruger
- Log-ind
- Fremsøgning af ønsket bog
- Reservationer
- Udlån
- Aflevering

Metodeovervejelser

Vi har i undervisningen benyttet et API til at indlæse data.
Vi gik denne vej til at starte med ift. at indlæse vores liste med bøger. Vi skiftede dog retning og lod vores router indlæse data fra databasen hver gang.
Her var vi i tvivl om, hvad der er 'best practice'. Vi valgte at gå med at indlæse data fra databasen, da dette synes mere medgørligt i forhold til, at vi tit havde brug for at indlæse data herfra. Det kunne dog også have været løst med at have et API med hver collection og så holde disse op mod hinanden og hvad der ellers måtte være behov for. API kræver kraft fra vores client-side, mens indlæsninger fra databasen kræver kraft server-side.

Vi valgte desuden at bruge Pug som template, da dette er ligetil at arbejde med, og det er det, vi har benyttet i undervisningen. Derudover var det som tidligere nævnt forudbestemt, at vi skulle benytte Node.js og Express.

Research

Det var et krav i projektbeskrivelsen, at brugerens password skulle være gemt som hashet værdi i databasen. Dette skulle gøres vha. 'brcypt' modulet, og det har vi researchet her: (*Bcrypt* n.d.), hvor vi har læst dokumentationen for at benytte det.

Vi havde brugt for at vores Mongoose schema havde unikke værdier, således den fx forhindrede at oprette dublikerede emailadresser. Her til har vi brugt følgende dokumentation: (*Mongoose v5.12.0: Validation* n.d.)

Vi har haft problemstillingen, at en bruger skulle logge ind og forblive logget ind. Det var nødvendigt for at kunne oprette et lån eller reservation samt se deres personlige detaljer herom. Her har vi undersøgt 'session' modulet: (*Express-Session* n.d.).

Vi har haft et behov for at søge på et array af værdier i vores MongoDB og hertil har vi brugt følgende dokumentation: (*$in — MongoDB Manual* n.d.).

Vi har haft behov for at sammenligne datoer for at tjekke både for sen aflevering som udløbet reservering. Her har vi brugt følgende (*Format JavaScript Date as Yyyy-Mm-Dd* n.d.) til at undersøge, hvordan vi formaterede datoformat samt sammenligning af datoer.

Analyse

Vi startede ud med at klarlægge projektet vha. wireframes. Dette gav et godt overblik over, hvordan applikationen skulle se ud og være struktureret samt få klarlagt, hvilke funktioner vi skulle have. Disse wireframes har vi sidenhen brugt, når vi skulle gå med næste step i projektet, og det fungerede rigtig godt at have en visuel plan.

Vi har en stor router til 'library'. Denne kunne muligvis have været mindre, og man kunne have valgt at dele den op i 'books', 'loans', 'reservation' og 'returns'. Der er dog meget her, der snakker sammen, derfor valgte vi at have det i en stor fil.

Vores handlers til samtlige collections tager sig dog af meget af det, hvorfor library-routeren stadig er overskuelig. Dog er 'router.get(/loansandreservations)' vokset til en del kodelinjer, og her kunne man godt slanke den ved at delegere noget ud til de forskellige handlers.

Optimalt bør routeren indlæse data og sende det til en handler, der modellerer dataen, og routeren så sender den modellerede data videre til siden.Dette har vi ikke været helt konsekvente med. Det er som oftest, når vi har flere modeller i spil, hvorved det har været svært at beslutte, hvilken handler der bør håndtere det.

Applikationen kunne have været udbygget mere med kategorier til bøgerne, der kunne bruges til fremsøgning (fx 'faglitteratur', 'skønlitterator' etc.).
Vi har ikke nået at lave en fremsøgningsmulighed, så listen af bøger vil være uoverskuelig, hvis den vokser. I en fremsøgning bør man kunne søge på både titel og forfatter.

Den personlige brugerside indeholder blot reservationer og udlån samt mulighed for at aflevere og låne reserveringer (hvis bogen er ledig).
Her kunne man med fordel have givet brugeren mulighed for at ændre deres personinfo samt at se indestående bøder. Det ville også være brugervenligt at give et overblik over, hvornår udlånene er foretaget samt afleveringsdato (30 dage efter udlån).

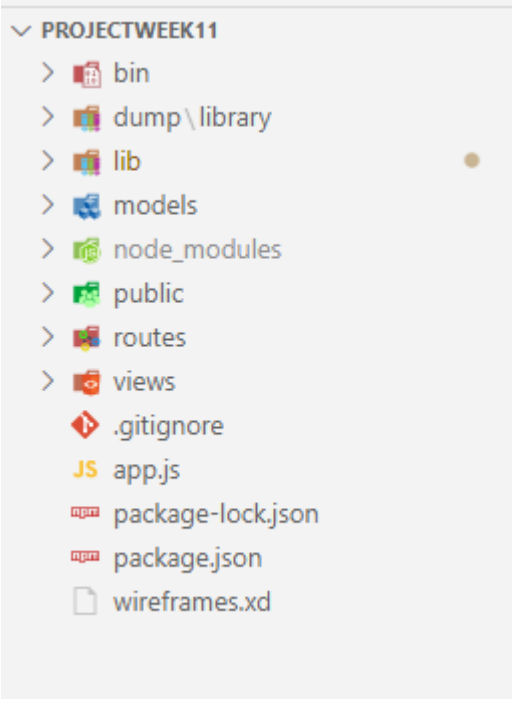
Vi stødte på størst problemer i vores funktion med at foretage et lån. Her havde vi stor udfordring med at finde frem til en ledig bog. Forinden brugeren kan foretage et lån, er der tjekket, hvorvidt der er en ledig bog, så dette kunne vi godt forudtage i selve funktionen med at oprette et nyt lån.
På nuværende tidspunkt tjekker funktionen for hver enkelt kopi af en bog op mod lån, og den sletter så lånet (blot fra det indlæste array), hvis der er match. Dette forudsætter dog, at arraysene med kopier og lån er sorteret ud fra id's, og det er de ikke nødvendigvis. Det kunne løses ved at sortere dem eller slette kopi i tilhørende array istedet. Således bliver denne ikke en mulighed at låne.

Konstruktion

Projektet indeholdt som nævnt et opstartet projekt, og vi har bibeholdt denne struktur. Vores modeller af ligger i 'models' mappen, hvor de har hver deres seperate mappe. Denne indholder en fil til schema og model og en handler-fil. Vores routere ligger i 'routes' mappen, og denne indeholder derfor kun disse tre filer.

I vores 'public'-mappe har vi vores stylesheet og i 'views' har vi vores pug-filer.

I vores 'lib' mappe har vi 'date.js' liggende, hvor vi har adskilt dato-funktioner. Her kunne vores 'mongooseWrap.js' også passende ligge, da denne indeholder database-funktioner.



Evaluering af proces

Vi har i dette projekt valgt at samarbejde om alt udarbejdelse af kode. Formålet med dette er at vi som gruppe opnår høj code ownership, samtidig med at vi hurtigere kan takle problemstillinger. For at vi kunne nå vores deadline uden uddelegering af arbejdsopgaver, har vi været meget grundige i vores planlægning. Til dette har vi anvendt et Kanban board (*Langehk/Projectweek11* n.d.). Her startede vi projektet ud med et planlægningsmøde, hvor vores features blev estimeret tidsmæssigt, og fordelt på ugens dage.
Kanban boardet er opdelt med kolonnerne ("To do", "In Progress" & "Done"). Hvilket var med til at strukturere hvilken feature vi arbejdede på, og samtidig hjælpe med, at vi kun havde en feature i "In Progress" ad gangen.

I vores sidste projekt savnede vi mere overvejelse af vores valg af metode og tilgang, inden vi begyndte at programmere. Dette har vi formået at lykkedes med i dette projekt, hvor vi har taget bevidste valg. Dette har resulteret i en langt bedre struktur, hvorved vi ikke har skulle bruge tid på at finde ud af, hvor hvad lå.

Konklusion

Vi opbevarer passwords som en hashet værdi, og de er dermed opbevaret sikkert. Vi sammenligner dog input-password i plaintext, da bcrypt har denne mulighed. Dette kunne optimeres yderligere ved at hashe input og så sammenligne.

Vores applikation viser bøgerne i en overskuelig liste, hvor brugeren kan vælge at gå ind og se detaljer på en underside til den specifikke bog og her låne eller reservere denne.
Vi har ikke nået at lave en mulighed for at fremsøge bøger, men i en optimal løsning skal dette dog være med.

Det kræver, at brugeren er logget ind, før man kan se undersiden til den specifikke bog, da man har brug for deres id til at oprette et lån eller reservering. Dette id får vi vha. session, der tager brugerens email-adresse med. Vha. vores 'Person model' har vi sørget for, at e-mailadressen skal være unik dvs. der kan ikke være duplikater heraf.

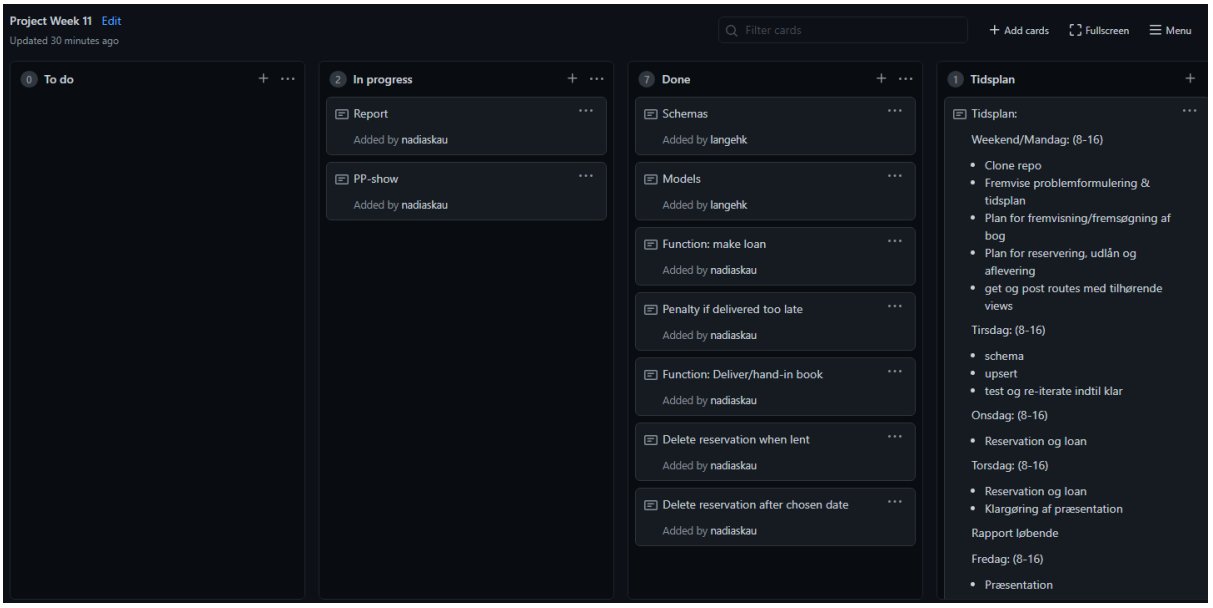
Når man kigger på den specifikke bog, har man muligheden for at låne den, hvis der er bøger tilgængelige. Dette tjekker vi på, idet siden indlæses, og derfor håndterer vores Pug-side, hvilket knap der skal vises: 'lån' eller 'reserver', hvor disse fanges af vores router, der håndterer lån eller reservering.

Når brugeren er logget ind, kan de på 'loansandreservations' se, hvilke lån de har samt hvilke reservationer. Her har de muligheden for at aflevere en bog, der går ind og sletter lånet i databasen. Når denne afleveres, tjekkes der op på, om bogen er afleveret for sent. Hvis dette er tilfældes, tildeles brugeren en bøde.
Deres reserveringer kan lånes, hvis bogen er blevet ledig. Dette tjekkes, når siden indlæses og Pug håndterer at vise knappen 'lån', hvis bogen er ledig. Hvis brugeren låner bogen, slettes reserveringen fra databasen og fremgår ikke længere af siden.

Applikationen indeholder samtlige funktioner, der er krævet i projektbeskrivelsen, og den er velfungerende. Hvis projektet havde haft en længere varighed, kunne man med fordel have brugt tid på at optimere brugervenligheden ift. styling samt flere funktioner.

Referencer

1.1 - Kanban board



$in — MongoDB Manual (n.d.) available from <https://docs.mongodb.com/manual/reference/operator/query/in/> [18 March 2021]
Bcrypt (n.d.) available from <https://www.npmjs.com/package/bcrypt> [18 March 2021]
Express-Session (n.d.) available from <https://www.npmjs.com/package/express-session> [18 March 2021]
Format JavaScript Date as Yyyy-Mm-Dd (n.d.) available from <https://stackoverflow.com/questions/23593052/format-javascript-date-as-yyyy-mm-dd> [18 March 2021]
Langehk/Projectweek11 (n.d.) available from <https://github.com/langehk/Projectweek11> [18 March 2021]
Mongoose v5.12.0: Validation (n.d.) available from <https://mongoosejs.com/docs/validation.html> [18 March 2021]