Prosjekt bibliotek

!DISCLAIMER: dette er tanker jeg hadde før jeg startet, så det er sannsynligvis utdatert og representerer ikke nettsiden som ligger på klasserom.net nå.

**Tanker før jeg har startet med arbeidet:**

Jeg har lyst til å lage et nettsted for online-bibliotek. Ideen bak er å kunne «låne» ebøker, men også å bestille bøker som man kan få levert på døren. På hovedsiden ser jeg for meg at man kan søke etter bøker og forfattere og filtrere etter sjanger, språk og vanskelighetsgrad. Det må også være en side med kontaktinformasjon. Jeg vil også at man kan opprette «lånekort» på nett, samt en brukerside tilknyttet lånekortet der man kan se hvilke bøker du har lånt og når du må levere dem tilbake. Jeg vil også ha en side der man kan legge inn spørsmål. Administratorene kan deretter legge inn svar på administratorsiden. Man må selvfølgelig også kunne legge inn nye bøker og merke boksider som inneholder feilinformasjon. Våre anbefalinger, hent ut brukerdata og sorter etter sjanger. Deretter de fire bøkene med flest likes innen de sjangrene.

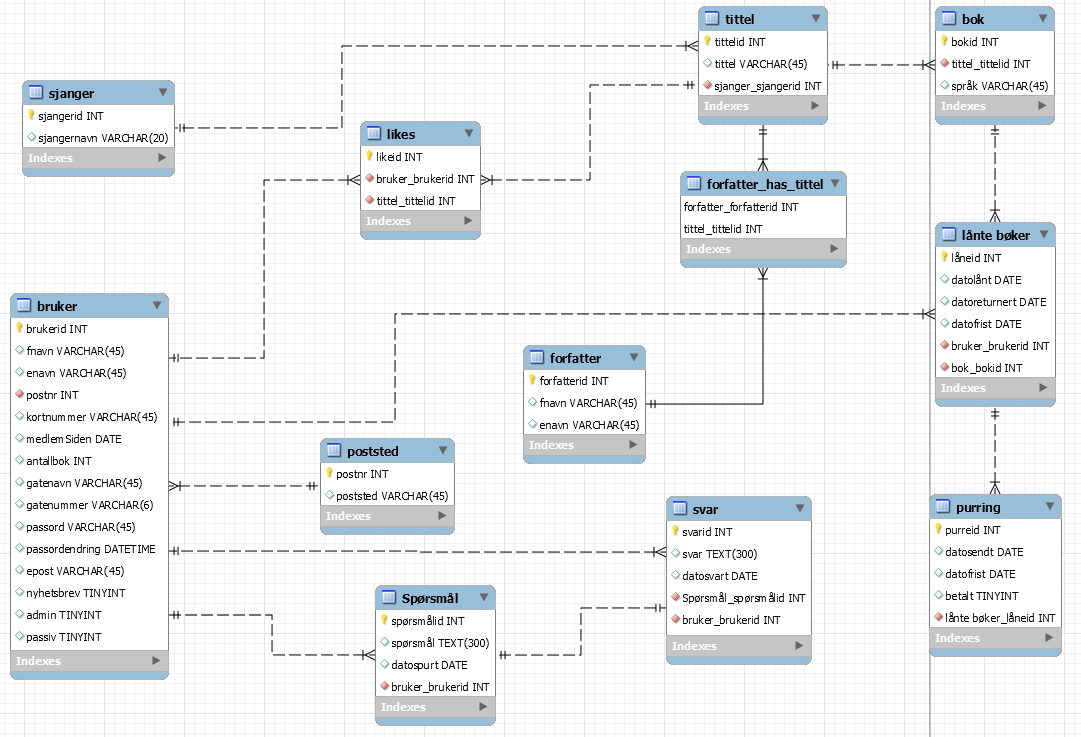
**Problemer jeg ser før jeg har startet:**

Dette blir et relativt stort og ambisiøst prosjekt, men jeg har jo ikke så mye bedre å gjøre for tiden så tid bør ikke være det største problemet. Jeg ser derimot for meg at det kan bli vanskelig å legge inn mange bøker, men jeg går ut ifra at jeg finner en løsning på dette. Om det blir at jeg får venner og familie til å manuelt legge inn bøker gjennom admin-siden eller om jeg finner en database med bøker som er open-source som jeg kan kopiere vet jeg ikke enda. Evt må jeg bare akseptere at det ikke blir så veldig mange bøker. Jeg skal i alle fall minst ha 50. Design (css og layout) blir nok også et lite problem for meg, men det så ikke ut til at det telte så veldig mye på dette prosjektet.

**Backlog:**

|  |  |  |
| --- | --- | --- |
| **DATO** | **Gjøremål** | **Logg** |
| 16.04.2020 | Planlegge hva jeg skal gjøre, lage en backlog, se på eventuelle problemer som kan dukke opp. | Fikk tenkt ferdig hva jeg har lyst til å gjøre med nettsiden min. Tenkt over problemer som kan dukke opp så jeg har det i bakhodet. Dette brukte jeg 2 timer på. |
| 17.04.2020 | Lage en grundig skisse for hvordan hovedsiden skal se ut i paint. Hvis jeg får tid begynne på HTML-koden. | Gjorde ferdig skisse for navigasjonsbar og footer, samt innholdet på hovedsiden. Startet også å strukturere div’er i html/css. Brukte 2 timer. |
| 20.04.2020 | Jobbe ekstra | Jobbet 8 timer med IT. Fikk laget ferdig hovedsiden sin html og css. Begynte også på side nummer to sånn smått. |
| 21.04.2020 | Lage skisser for resten av sidene. Lage HTML-struktur for de. | Laget ferdig HTML-struktur for de resterende sidene. |
| 24.04.2020 | Gjøre datamodellering | Fikk gjort ferdig datamodellering og tenkt litt mer gjennom det praktiske ved siden. |
| 25.04.2020 | Jobbe med innhold til sidene |  |
| 28.04.2020 | Jobbe med innhold til sidene |  |
| ~~01.04.2020~~ | Jobbe med innhold til sidene |  |
| 05.05.2020 | Jobbe med innhold til sidene |  |
| 08.05.2020 | Fikse bugs |  |
| 12.05.2020 | Skrive prosjektbeskrivelse |  |
| 15.05.2020 | Skrive prosjektbeskrivelse |  |

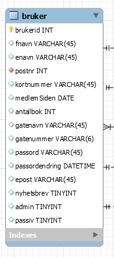
Jeg førte ikke logg etter 24. april. Dette fordi jeg har jobbet til sent på kveld nesten hver dag og ikke tok meg tid til å føre logg klokken 11 på kvelden.

Datamodellering: 

**Forklaring på datamodelleringen:**

*- Atomærkravet*

For at en database skal være så effektiv som mulig er det viktig at vi lagrer verdiene våre på best mulig måte. En ting man kan gjøre er å dele opp informasjonen så mye som mulig. Dette betyr for eksempel at vi lagrer fornavn og etternavn i to forskjellige kolonner. Regelen om å dele opp informasjonen så mye som mulig kalles atomærkravet. Jeg skal nå gå gjennom min database og vise hvorfor den oppfyller atomærkravet. Jeg legger ved skjermdumper av tabellene jeg går gjennom så du slipper å skrolle til toppen av dokumentet. Jeg kommer til å gå gjennom alle tabellene nedenfor.

**Tabell Bruker:**

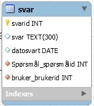
Merk at kolonnene passordendring, kortnummer, nyhetsbrev og antallbok ikke er i bruk. Kortnummer og antallbok har jeg slettet fra den nyeste datamodellen min. Kolonnen passordendring hadde jeg tatt i bruk dersom jeg hadde hatt mer tid. I tabellen ser vi at navn er delt inn i fornavn og etternavn for å oppfylle atomærkravet. Adresse er delt opp i gatenavn og gatenummer for å oppfylle atomærkravet. Postnr er fremmednøkkel til tabellen poststed. Dette er gjort slik at brukeren ikke har mulighet til å skrive inn feil poststed og slik at vi oppfyller tredje normalform, som jeg skriver om på side 7/8. Informasjon om passord skulle jeg ha kryptert dersom jeg hadde mer tid, men det har jeg ikke gjort enda. I den nyeste modellen har jeg også en kolonne som heter «telefon», det er denne man bruker for å logge seg inn. Datatypene er valgt etter hva som virket logisk, men jeg skal forklare de datatypene som kanskje ikke kan forstås like intuitivt. Gatenummer skulle tilsyneslatende vært integer, men ettersom man også kan ha bokstaver i gatenummeret så satte jeg den til varchar. F.eks 111B. De nederste tre; nyhetsbrev, admin og passiv har datatype «TINYINT». Denne datatypen lagrer informasjon som boolske variabler, altså TRUE OR FALSE. True lagres som 1, mens false lagres som 0.

**Tabell poststed:**

Tabellen poststed er ganske liten, så jeg skal ikke skrive så mye om den. Det jeg vil si er at hvert postnummer, som er unikt, har et poststed knyttet til seg. I databasen min fikk jeg til å importere alle postnummer i hele Norge og poststedet som er knyttet opp mot det fra et excel-dokument publisert av Posten Norge inn i databasen min. Dette gjorde jeg via en annen nettside som konverterte excel til sql kode. Det var selvfølgelig noen feil i koden som ble laget, så jeg brukte en del tid på å fikse dette manuelt. Sluttproduktet er at man kun trenger å skrive inn postnummer, også vet admin hva poststedet er.

**Tabell spørsmål:**

Tabellen spørsmål innholder heller ikke så mye informasjon, men den har informasjon om hva spørsmålet er, hvilken dato det ble spurt, og hvem som stilte spørsmålet. Noe som er verdt å merke seg er at det er en 1-1 kobling mellom tabellen spørsmål og svar. Det skal jeg forklare under.

**Tabell svar:**

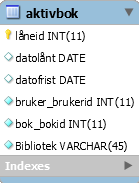
Tabellen svar inneholder i bunn og grunn samme informasjon som spørsmål, bare at det lagres et svar fremfor et spørsmål. I tillegg lagres det informasjon om hvilket spørsmål dette er et svar på. Årsaken til at jeg flyttet svaret til en egen tabell er at ikke alle spørsmål blir besvart, og det er ugunstig å lagre mange tomme felt i en tabell. Det er kun administratorer som kan sende inn svar, og det er kun ett svar per spørsmål.

**Tabell likes:**

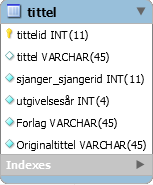
Selv om tabellen likes ikke er i bruk vil jeg gjerne forklare tanken bak den uansett. Tabellen fungerer slik at en bruker kan like en bok, og da opprettes det en rad i tabellen likes med informasjon om hvem som likte og hvilken bok som ble likt. Den er satt i en egen tabell ettersom den fungerer som en mange-til-mange kobling mellom bruker og tittel ettersom flere brukere kan like en tittel, og en bruker kan like flere titler. For å få ut hvor mange likes en bok har spør man bare etter hvor mange rader det er i tabellen likes hvor kolonnen tittel\_tittelid er lik tittelid. Brukeren kunne også hatt muligheten til å se hvilke bøker hen har likt med å spørre etter hvor mange rader som finnes i tabellen likes hvor bruker\_brukerid er lik brukeriden til brukeren.

**!DISCLAIMER**: skjermdumpene fra tabellene nedenfor er tatt fra en eksport av tabellstrukturen på mysql.klasserom.net. Den er litt dårlig, så oppmerkingen på fremmednøkler er ikke tatt med. ALLE kolonner som har navn med understrek «\_» i er fremmednøkler. De er fremmednøkler med utgangspunkt i tabellnavnet som står før «\_».

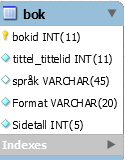
**Tabell lånte bøker (nytt navn «aktivbok»)**

Tabellen «aktivbok» inneholder informasjon om de bøkene som er til utlån akkurat nå. Når en bok blir lånt opprettes det en låneid med tilhørende informasjon, og når den blir levert tilbake slettes låneiden og informasjonen. I ettertid kunne jeg sett for meg at det hadde vært lurere å legge til en boolsk variabel «aktiv» som sa om boka var til utlån akkurat nå eller ikke. Dette for å lagre informasjon om bøker som har blitt lånt ut tidligere også. Dessverre tenkte jeg ikke over denne muligheten når jeg lagde datamodellen. Tabellen er koblet opp mot tabellen bruker og bok. Kolonnen «bibliotek» sier noe om hvilket bibliotek denne boka er lånt fra, ettersom man kan bestille bøker fra landets 10 største biblioteker.

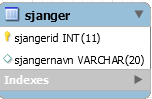
**Tabell tittel**

Tabellen tittel inneholder informasjon om en gitt tittel, med informasjon om hva tittelen heter, utgivelsesåret, forlaget og kobling til tabellen sjanger. Jeg har delt opp bøkene i «tittel» og «bok» ettersom man kan registrere flere eksemplarer av samme tittel, og eksemplarene kan ha forskjellig språk, format og sidetall.

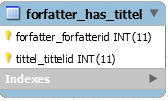
**Tabell bok**

Tabellen bok inneholder informasjon om hvert enkelt eksemplar som finnes av en tittel. Som sagt i avsnittet over inneholder denne tabellen informasjon om språket, formatet og sidetallet. Når en administrator skal registrere et eksemplar velger hen en tittel fra en liste hen kan søke i.

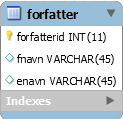
**Tabellen sjanger**

Tabellen sjanger inneholder en primærnøkkel-kolonne «sjangerid» og en kolonne sjangernavn som sier noe om hvilken sjanger det er. Sjanger kunne egentlig vært satt opp under «tittel», men jeg valgte å dele det opp så mye som mulig slik at administratoren kan registrere flere sjangre fremfor at listen satt fast i html-koden. Denne avgjørelsen stemmer også overens med atomærkravet.

**Tabellen «forfatter\_has\_tittel»**

Denne tabellen inneholder bare informasjon om koblingen mellom forfatter og tittel. Det er en egen tabell ettersom koblingen mellom tittel og forfatter er en mange-til-mange kobling. Årsaken til dette er at en forfatter kan ha skrevet flere bøker, og en bok kan ha flere forfattere. Hvis den informasjonen ville vært lagret i samme tabell ville det brutt med atomærkravet. Dette er det et eksempel på i boka på side 141 Tabell 13.4.

**Tabell forfatter**

Denne tabellen inneholder informasjon om hver enkelt forfatter. Den følger atomærkravet ettersom navnet er delt inn i fornavn og etternavn. I ettertid så tenker jeg at det kunne vært lurt å lagre mer informasjon om vher enkelt forfatter. For. eksempel kjønn, fødselsdato, stedet de er fra, nasjonalitet, etc etc. Det komprimerer derimot ikke funksjonaliteten til nettsiden min at jeg ikke har denne informasjonen, ettersom brukeren ikke har alternativet å lese om forfatteren bak en bok. Kanskje kunne jeg lagret en link til wikipedia-artikkelen til hver enkelt forfatter.

-*Normaliseringsformene*

Nå er jeg ferdig med å forklare innholdet i hver enkel tabell, og hvorfor de følger atomærkravet. Jeg skal også forklare hvorfor datamodellen min følger normaliseringsformene. Det kommer nå. Først og fremst følger de 1. NF ved at alle tabeller oppfyller atomærkravet og har en primærnøkkel. Andre normalform (2. NF) er oppfylt ved at første normalform er oppfylt og ingen kolonner kan bestemmes av deler av primærnøkkelen. Dette betyr at vi ikke kan hente ut samme informasjon fra flere tabeller. Et eksempel på brudd på andre normalform ville vært om jeg hadde informasjon om både sjangeren og sjangeriden i tabellen «tittel». Tredje normalform er oppfylt ved at første og andre normalform er oppfylt og at ingen av kolonnene bestemmes av noe annet enn primærnøkkelen. Et eksempel på brudd på dette er hvis jeg hadde hatt både kolonnen poststed og postnummer i tabellen «bruker», ettersom poststedet kan bestemmes av postnummeret. Ettersom dette ikke er tilfelle i noen av tabellene mine kan vi si at databasen min er normalisert fordi 3. NF er oppfylt i alle tabeller.

-*Koblinger*

En sjanger kan være knyttet til flere titler, men en tittel er bare knyttet til en sjanger. Dette er selvfølgelig diskutabelt, men jeg velger å holde det «enkelt» og kun ha en sjanger. Det er videre flere eksemplarer av samme tittel, men et eksempler kan bare ha en tittel, derfor er det en en-til-mange kobling mellom tittel og bok. En bok kan ha blitt lånt ut flere ganger, men en låneid kan bare være knyttet opp til en bok. Igjen så kan det være flere purringer på samme låneid, men en purreid tilsvarer bare en låneid. Mellom tittel og forfatter er det en mange til mange kobling fordi flere titler kan være skrevet av en forfatter, og flere forfattere kan ha skrevet en bok. En brukerid kan svare til flere spørsmål, ett spørsmål kan bare ha ett svar. Igjen så kan et spørsmål bare være besvart av en administrator, og en administrator kan svare på flere spørsmål. Derfor en til mange mellom bruker og spørsmål, en til en mellom spørsmål og svar, og en til mange mellom bruker og svar. Til slutt så kan en bruker bare være knyttet til et postnummer, og et postnummer kan være knyttet til flere brukere. Poststed er i samme tabell som postnummer fordi dersom man har postnummeret så kan man finne poststedet i registrene til posten.

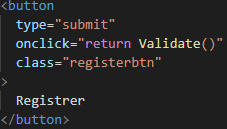
# Forklaring av kodebiter og funksjonalitet.

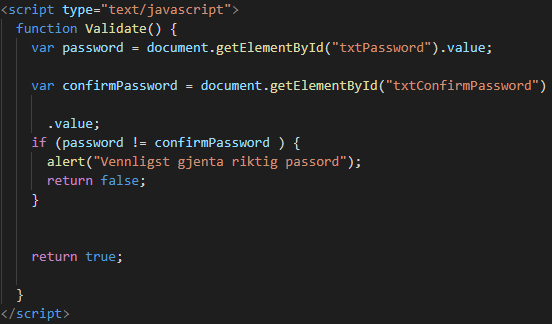
Her kommer jeg til å skrive om hvordan kodestrukturen min er, hvordan innloggingsfunksjonen, registreringsfunksjonen, navbaren, brukeroversikten, søkefunksjonen, spørsmåls- og svarfunksjonen og passordendring og sletting av bruker fungerer.

**-Registrering av bruker.**

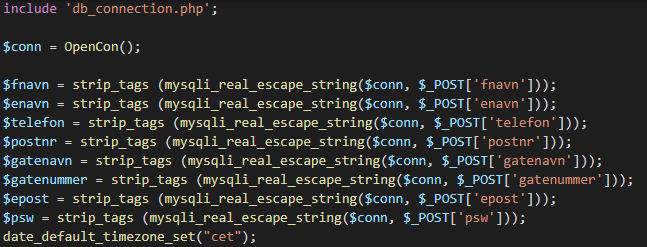


\*HTML INPUT FIELDS\*

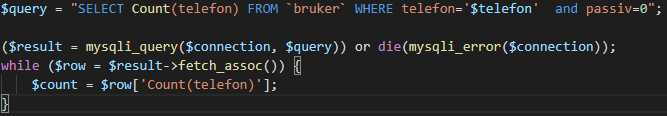




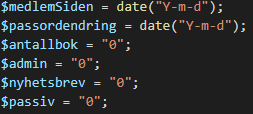
Her ser du grovt sett koden for registreringskjemaet. Javascripten sørger bare for at brukeren har skrevet inn samme passord to ganger på rad, slik at man unngår tastefeil. Dersom passordet er gjentatt feil vil formen ikke bli sendt og det dukker opp en alert. Dersom det gjentatte passordet er riktig blir man sendt videre til en action page. Du finner grovt sett koden for den nedenfor.



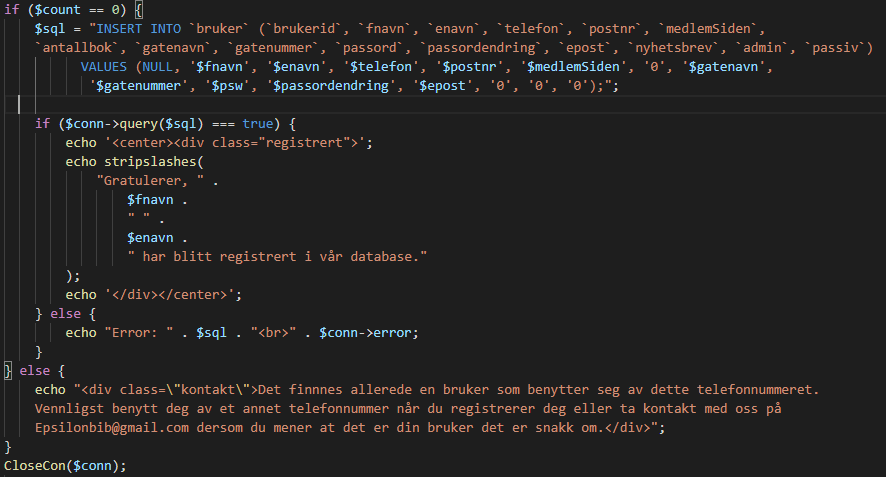
Først inkluderer jeg en connection page som kobler oss opp mot databasen vi arbeider med. Etterpå definerer vi de variablene som ble sendt inn av skjemaet fra tidligere. Jeg har lagt til php funksjonene strip\_tags og mysqli\_real\_escape\_string slik at tullete folk ikke kan prøve å SQL injecte meg eller skrive inn html-kode i input fieldsene. Dette la jeg til ettersom det var noen fra klassen som prøvde å ødelegge databasen min med å sende inn såkalt «malicious code». Fordelen er at jeg lærte en del om datasikkerhet av dette.



Videre gjør vi en SQL-spørring som sjekker om det er noen andre brukere som er registrert med samme telefonnummer. Spørringen spør etter hvcor mange rader som inneholder det telefonnummeret brukeren har skrevet inn. Hvis spørringen gjennomføres, blir variabelen $count definert.



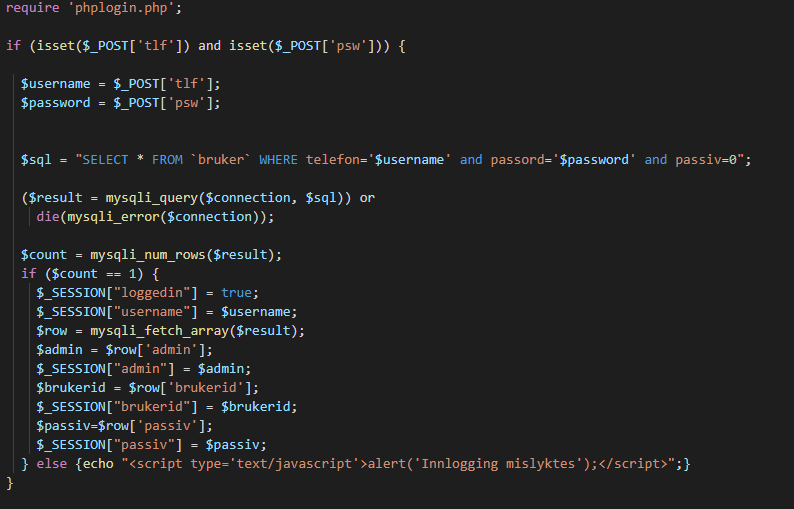
Etterpå definerer vi resten av variablene som skal settes inn. Jeg kunne vært lur og satt standaren på kolonnene som skal være 0 til å være 0. Det tenkte jeg ikke på.

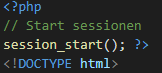


Så sier jeg at dersom det ikke er registrert noen brukere med samme telefonnummer så opprettes brukeren. Dersom det er registrert en bruker med samme telefonnummer så blir det sendt tilbake en feilmelding. En endring jeg gjerne kunen gjort her er å hashe og salte passordet før det blir sendt til databasen, men det satte jeg meg ikke inn i. Dessuten har jeg sikret det nok ved å stoppe SQL injections og gjøre slik at admins ikke kan se passordet til hver enkelt bruker.

-**Innlogging og sessions**

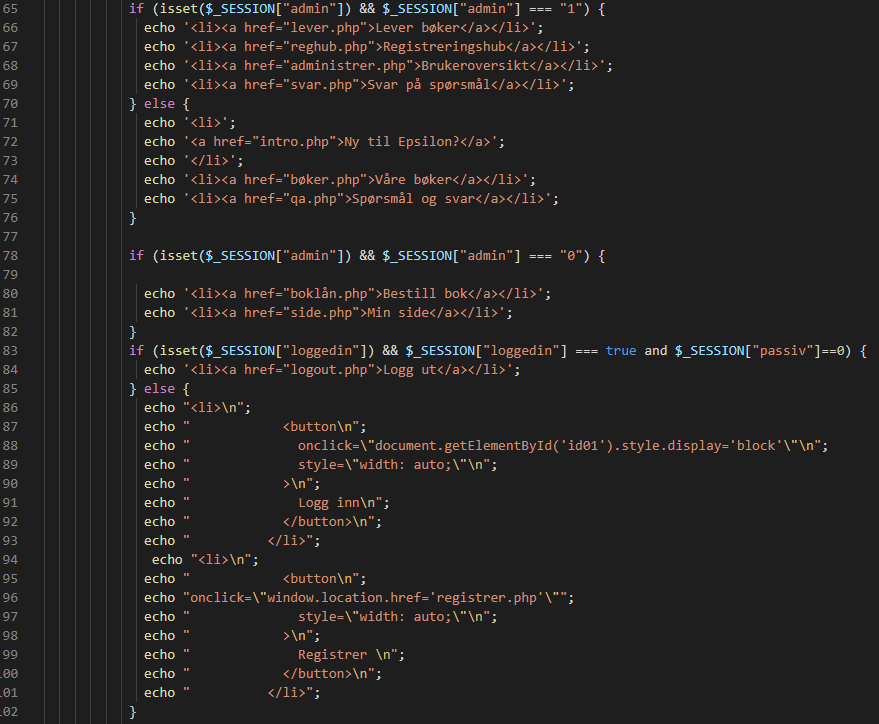
På alle sidene er det en innloggingsmodal som dukker opp dersom du klikker på knappen «logg inn». Når man trykker på knappen dukker det opp et skjema der man logger inn eller trykke på registreringslinken og registrerer seg. Action pagen på skjemaet som dukker opp er index.php. Koden for verifiseringen ser slik ut:



Her ser du at hvis man har sendt inn innloggingsinfo så defineres variablene $username og $password. Videre gjøres det en SQL-spørring som sjekker om det finnes en rad som inneholder det innsatte telefonnummeret/brukernavnet og passordet. Den sjekker også om den boolske variabelen «passiv» er satt til 0. Dette er for å gjøre slik at man ikke kan logge inn hvis brukeren har blitt deaktivert. Du kan også se at vi lagrer diverse variabler som en session key. Dette gjør at vi kan hente ut informasjon om hvem som har logget inn, og at man ikke trenger å logge inn på nytt hver gang man går til en annen side. Det gjør også at man holder seg innlogget dersom man ikke logger ut. For at dette skal funke pmå man inkludere denne kodesnutten i starten av hvert dokument:

**-Navbar laget i php**

Hele Nav-baren min er laget i php, dette er fordi jeg vil at den skal forandre seg dersom det er en bruker eller admin som logger seg inn. Som sagt over så lagrer jeg diverse variabler som session keys, dette betyr at jeg kan kjøre if-setninger som spør om man er logget inn som admin, bruker eller ikke logget inn i det hele tatt.

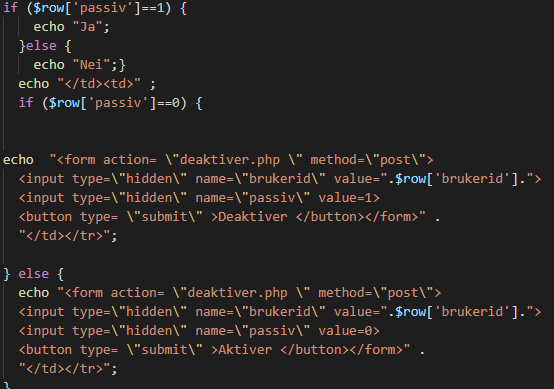


Her ser du at hvis man er logget inn og logget inn som admin så vises linken til de fire admin-funksjonene. Ellers så vises det innholdet som ikke er bruker-sensitivt. Videre sier koden at dersom man er logget inn og logget inn som bruker så vises «bestill bok» og «min side». Ellers så vises logg inn og registreringsknappen. Jeg har brukt css til å style nav-baren med bruk av flexbox, men den koden skal jeg ikke inkludere her.

**-Brukeroversikten**



Brukeroversikten fungerer slik at den først sjekker om du er logget inn som admin, og hvis du er det kan du se på oversikten. Det blir echo-en tabell som inneholder informasjon om hver bruker samt et søkefelt som gjør at du kan filtrere brukere. Koden der er ganske basic, så jeg viser ikke den. Det som derimot er litt gøy er deaktiver-knappen og de boolske variblene som sier om brukeren er meldt på det fiktive nyhetsbrevet og om brukeren er deaktivert.



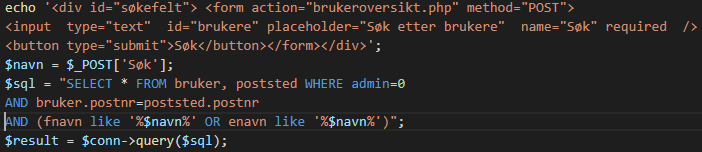
Her ser du at jeg har gjort om boolske variabler til Ja/Nei for at det skal være lettere å skjønne. Knappen varierer også slik at den kan både aktivere og deaktivere en bruker. En funksjon jeg gjerne skulle lagt til er at admin kan redigere informasjonen om hver enkelt bruker. Dette kunne jeg gjort med å lage et redigeringspanel som dukker opp når man trykker på en bruker.

**Søkefunksjon i brukeroversikt og bokoversikt.**

Jeg har også en søkefunksjon som lar administratoren søke blant brukere. Koden for den ser slik ut:



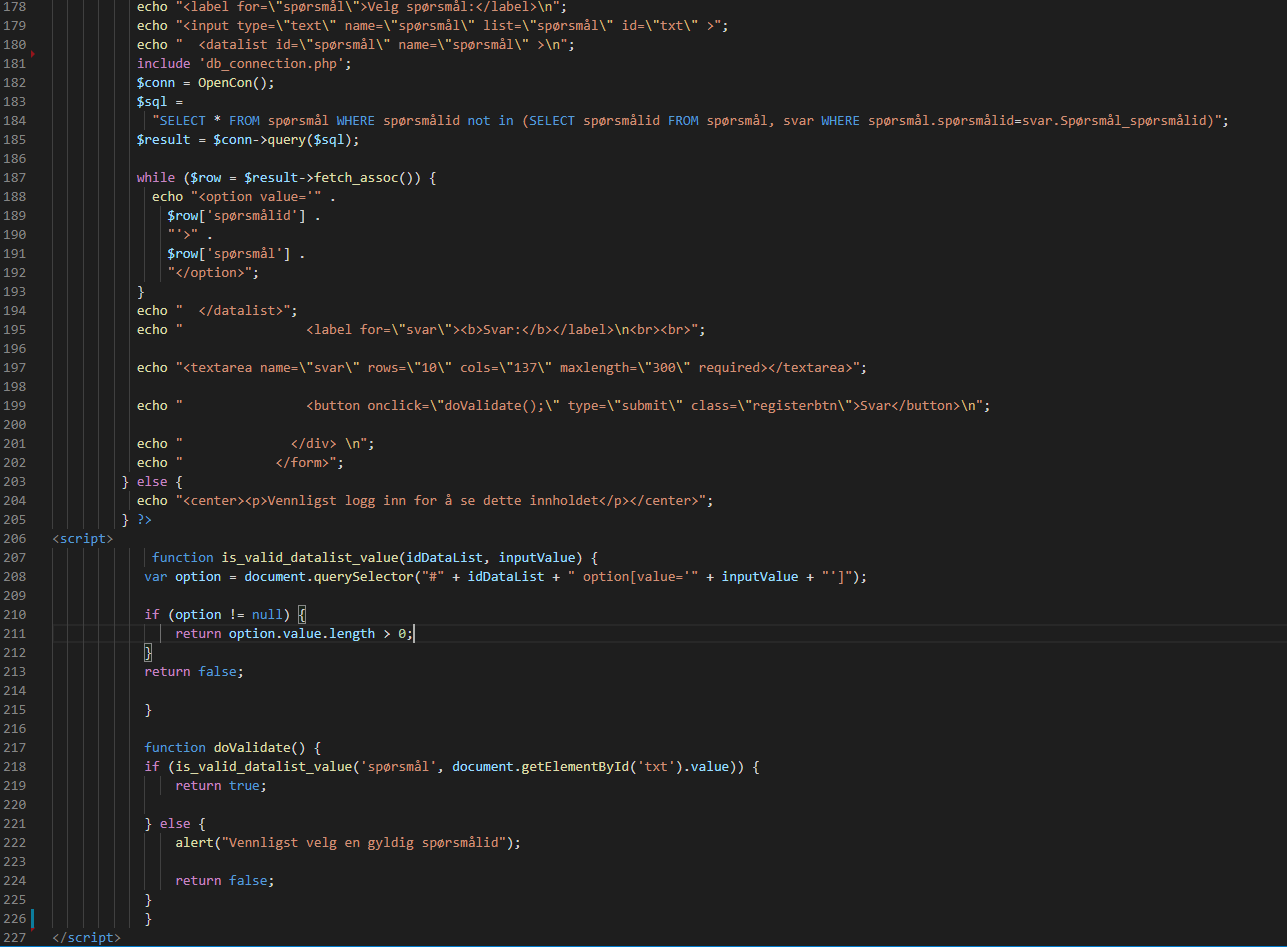
Når man trykker på søk blir man sendt til et nytt dokument som viser resultatet fra søket med de samme funksjonene som i første brukeroversikten. Hadde denne databasen arbeidet med massive mgnder brukere ville jeg satt en limit på antall brukere som blir vist og gjøre så man kan bytte side ofor å se flere. Koden for resultatsiden ser slik ut:



Her ser du at vi gjør en SQL-spørring som leter etter likheter fra søkefeltet i kolonnen navn og etternavn. En umiddelbar svkhet som kanskje du også ser er at man må søke på ENTEN fornavn ELLER etternavn, man får ingen resultater dersom man søker etter fornavn + etternavn. Dette kunne jeg også ha fikset med mer tid eller hvis jeg var mer effektiv. Outputen av denne vises i en tabell med samme struktur som i brukeroversikten. Jeg kommer ikke til å ta for meg søkefunksjonen for bøker, ettersom den baserer seg på akkurat samme prinsipp.

-**Spørsmål og svar.**

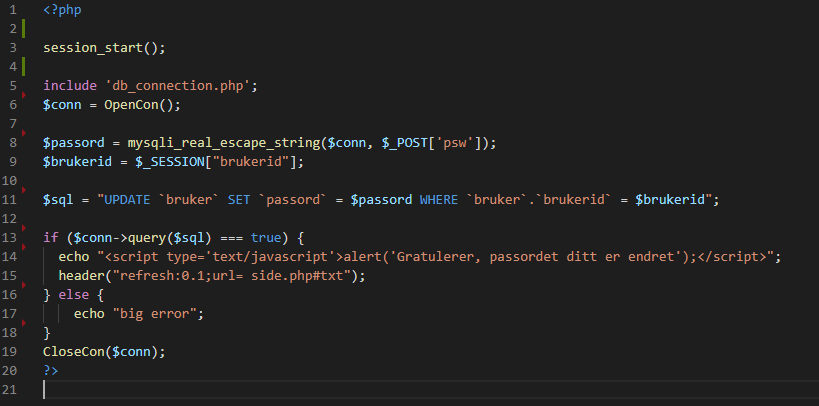
Når man sender inn et spørsmål blir spørsmålet og hvem som stilte spørsmålet lagret. Når en admin skal svare på et spørsmål velger hen det spørsmålet hen vil svare på og skriver et svar til det. Da lagres det informasjon om hvilket spørsmål som ble besvart, svaret og hvem som sendte svaret. Koden bak det er ikke veldig fancy, men jeg vil vise koden bak dropdown-menyen som admin får opp når hen skal velge spørsmål å svare på. Der har jeg nemlig brukt en <datalist> tag fra html sammen med php og en dæsj javascript. Du kan se koden under.



Her ser du et input field som er tekst, og som er koblet opp mot en dataliste med verdier fra en sqlspørring som spør etter ubesvarte spørsmål. Du kan begynne å skrive inn et spørsmål, velge spørsmålsiden og skrive et svar til den spørsmålsiden. Javascripten du ser bekrefter bare at spørsmålsiden du skriver inn, finnes i spørringen som ble gjort tidligere. Denne typen datalist har jeg brukt i flere dokumenter, så jeg forklarer den ikke noen flere ganger.

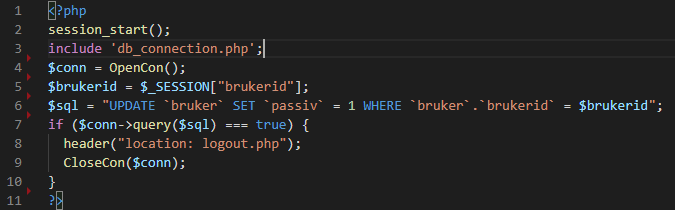
**-Passordendring**

Som et ekstra sikkerhetstiltak har brukerne mulighet til å bytte passord fra siden som heter «Min Side». Her skriver man inn det nye passordet to ganger også blir man sendt til en connectionpage. Her er koden for den siden man blir sendt til:



Her ser du at vi oppdaterer passordet til den brukeren som er logget inn. Hvis spørringen blir gjennomført og passordet blir oppdatert kommer det en javascript alert som sier at passordet er endret. 0,1 sekunder etter at man har lukket alerten blir man sendt tilbake til side.php omtrent samme sted som du forlot den fra. Dette er gjort med en header som har delay på 0.1 sekund og mål side.php.

**-Slett bruker**

Når en bruker sletter brukeren sin, blir brukeren egentlig bare deaktivert. Dette er for å løse problemet med at en bruker sletter dataen sin etter å ha lånt en bok. Dessuten bryter det ikke med EU’s GDPR reglement. Koden finner du nedenfor: 

Her ser du at vi oppdaterer bare kolonnen ‘passiv’ til å være 1. Altså blir brukeren deaktivert og den kan ikke lenger logge inn.

THE END!!!