NTNU

Kunnskap for en bedre verden

Bacheloroppgave

IE303612 Bacheloroppgave

Kandidatnummer:

10003

Totalt antall sider inkludert forsiden: 85

Ålesund, 30.11.18

NTNU

Kunnskap for en bedre verden

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	☒
2.	**Jeg/vi erklærer videre at denne besvarelsen:** • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☒
3.	**Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§14 og 15.**	☒
4.	**Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver**	☒

5.	**Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens studieforskrift §31**	☒
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	☒

Publiseringsavtale

Studiepoeng: 20

Veileder: Arne Styve

NTNU i Ålesund

Bacheloroppgave

Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven §2).

Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage HiM med forfatter(ne)s godkjennelse.

Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved NTNU i Ålesund en vederlagsfri rett til å

gjøre oppgaven tilgjengelig for elektronisk publisering: ☒**ja** ☐**nei**

Er oppgaven båndlagt (konfidensiell)? ☐**ja** ☒**nei**

(Båndleggingsavtale må fylles ut)

- Hvis ja:

Kan oppgaven publiseres når båndleggingsperioden er over? ☐**ja** ☐**nei**

Er oppgaven unntatt offentlighet? ☐**ja** ☒**nei**

(inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)

Dato: 30.11.18

Forord

Jeg valgte denne oppgaven fordi den virket interessant. Å lage en web applikasjon basert på et av de nyeste og mest relevante bibliotekene i dag virket som en veldig relevant erfaring å ta med seg inn i jobbmarkedet, samtidig som det er et spennende felt å utvikle noe i. Det bygger samtidig noe på de tidligere fagene vi har vært i gjennom; Datakommunikasjon med nettverksprogrammering, Datamodellering og databaseapplikasjoner, Systemadministrasjon og Webteknologi. Gjennom bruk av Javascript, HTML, CSS, kommunisere over Ethernet og database.

Oppdragsgiver er Cordel Norge.

Jeg vil gjerne takke:

Arne Styve for god hjelp og veiledning gjennom prosjektet.

Ole Johan Larsen og Harald Bjørshol hos Cordel Norge for god hjelp og lærerikt samarbeid gjennom prosjektperioden.

Sammendrag

Flere og flere i dagens samfunn finner ut at en web løsning for å håndtere noe vil være en forbedring. Utviklingen til en webapplikasjon har vært enorm og i dag har du muligheten til å lage nesten hva som helst på en webapplikasjon og i tillegg så kan brukergrensesnittet være slik at hvem som helst kan bruke de mest komplekse systemer.

Målet med prosjektet er å effektivisere arbeidsdagen til de utviklerne som løser feilmeldingene som kommer inn hos Cordel. Cordel har flere forskjellige produkter som brukes over hele landet og snart Skandinavia. Ingen system er feilfrie og da ligger det mye på hvordan feilene håndteres og hvor raskt de håndteres. Dette prosjektet skal gjøre det enkelt for utvikleren å få en god oversikt alt av informasjon om feilmeldingen, samtidig som den skal kunne endre statusen til en feilmelding når den er ferdig håndtert.

Denne rapporten gir grunnleggende informasjon og teori om hva som liger bak de løsningene jeg har valgt å bruke i denne web applikasjonen. Blant annet så har jeg brukt React som et bibliotek for klient programmeringen, der Javascript sammen med HTML og CSS har blitt brukt for å gjøre nettsiden visuelt tilstrekkelig og brukervennlig for alle utviklerne på Cordel. PostgREST som automatisk lager sluttpunkter av dataen i postgreSQL databasen. Og da Swagger som gjør det enkelt for meg å håndtere endepunktene PostgREST lager til meg.

Sluttproduktet av dette prosjektet skal være enkelt å bygge videre på og utvide om det skulle være ønskelig. Klient siden er satt opp med forskjellige komponenter, der en fint kan bygge videre på uten at det skal være nødvendig å endre noe nevneverdig på det som allerede er der fra før.

Terminologi

IDE – Integrated development environment

API - Application Programming Interface

REST - Represntational State Transfer

JSON - JavaScript Object Notation

SQL - Strucktured Query Language

SPA – Singel-page application

JS – Javascript

HTML – Hypertext markup language

CSS – Cascade style sheet

AJAX - Asynchronous JavaScript and XML

URI – Uniform Resource Identifier

TCP/IP – Transport Control Protocol /Internet Protocol

URL – Uniform Resource Locator

GUI – Graphical user interface

UML – Unified Modeling Language

NPM – Node Package Manager

Innhold

Figur liste

NTNU i Ålesund

Bacheloroppgave

1. Introduksjon

Denne bacheloroppgaven er skrevet ved NTNU Ålesund i samarbeid med Cordel Norge. Cordel Norge er et utviklerfirma som leverer systemer til kunder over hele landet. De har en lokal bedrift i Ålesund, og flere ansatte spredd utover i Norge. De har i flere tiår jobbet med systemer som skaper kostnadsreduksjon, effektivisering og å gjøre bedrifter konkurransedyktige. I dag er Cordel i ferd med å gå til SAAS og utvide til resten av Skandinavia. De har også over 10 000 brukere over hele landet. [36]

Målet med denne bacheloroppgaven er å utvikle en web applikasjon som er basert på de nyeste og beste metodene som tilgjengelig i dagens samfunn. Web applikasjonen skal inneholde et system som gjør det lettere for Cordel å ha kontroll og håndtere feilmeldingene som kundene deres evt. Måtte ha. Skal være mulig å søke i de forskjellige feilmeldingene, søke i de forskjellige kundene og kunne sette status på dem.

Etter de første møtene med Cordel, så ble det brukt en del tid på å avgjøre hvilket rammeverk, programmeringsspråk og hvilke utviklingsmetoder som ville passe best for web applikasjonen som skulle bli laget.

Denne bacheloroppgaven vil være strukturert slik at leseren får først innsikt om hvorfor de forskjellige hjelpemidlene ble brukt og hvilke andre alternativer som kunne vært aktuelle. Deretter vil det bli en forklaring om hvorfor jeg har løst oppgaven slik som jeg har gjort. Etterfulgt av en drøfting om hva som kunne blitt gjort annerledes og bedre.

Jeg vil forvente at leseren har en basis forståelse om programmeringsutvikling og objekt orientert programmering. Disse forutsetningene har blitt tatt når oppgaven ble skrevet. Det er også et krav om at leseren har en viss forståelse for engelske ord og uttrykk.

1.1 Begrensninger

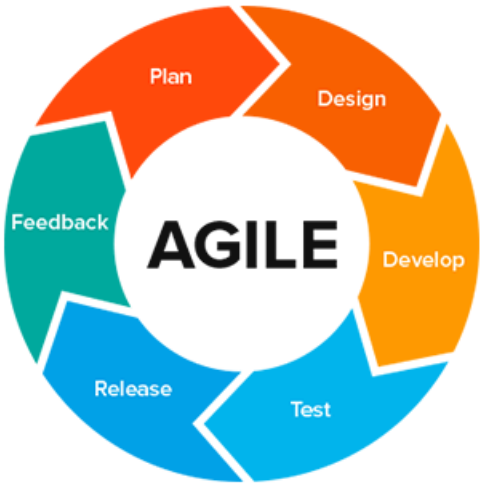
Begrensningene for å få applikasjonen helt ferdig er tidsbruken. Mye tid vil gå til undersøkelser av de riktige fremgangsmetodene og hvordan prosjektet skal gjennomføres på best mulig måte.

2. Teori

2.1 Agile metoder

Agile metoder I en software utvikling vil si å korte iterasjoner der kunden/brukeren blir mye inkludert. I software utvikling så blir det ofte brukt program slik som JIRA eller lignende for å skape en struktur på alle målene som skal gjøres. Det blir opprettet en sprint for hver iterasjon som gjøres og teamet blir tildelt oppgaver.

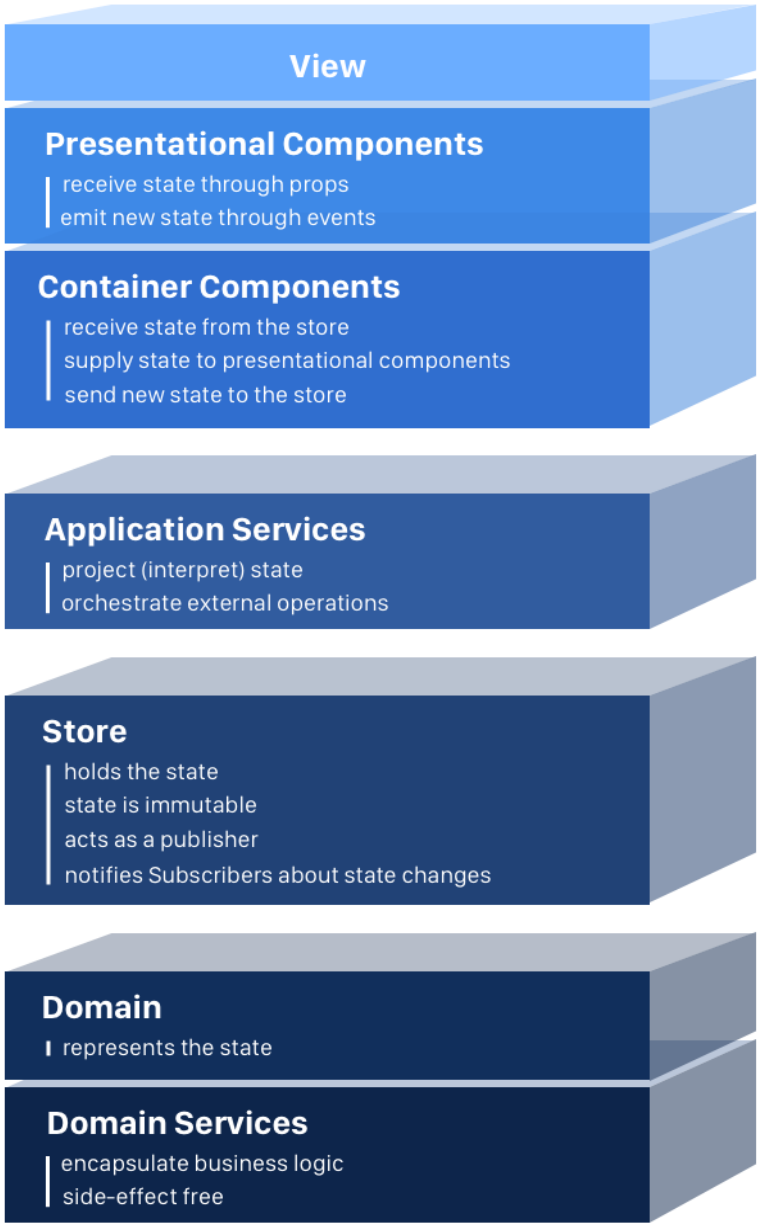
Poenget med å bruke agile metoder er for å kontinuerlig skape fremgang, samtidig som det fremtvinger en fleksibilitet i prosjektet der det kommer raske endringer i forhold til hva brukeren/kunden ønsker



.

2.2 Webapplikasjonens oppbygging

En webapplikasjon er ofte bygd opp av 4 store lag eller «layers» som en bruker mest innenfor programmering:



Figur 1 Oppbyggingen til en webapplikasjon[29]

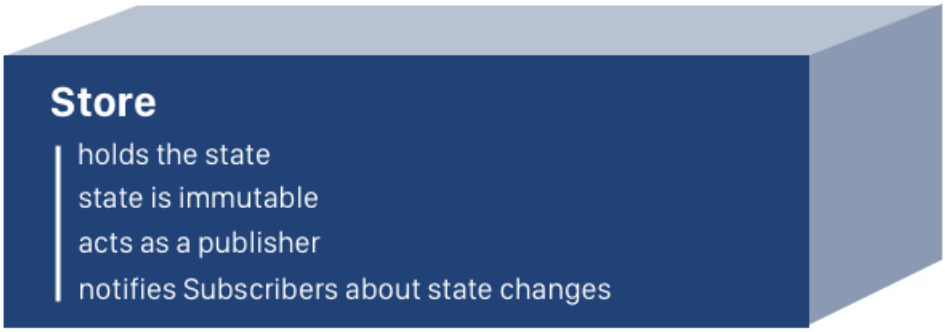
2.2.1 The domain layer

Domain'en sin oppgave er å beskrive hvilken state applikasjonen er i og samtidig holde på «business logic»'en også noe som vil si å holde kommunikasjonen mellom en sluttbruker og en database i gang og opprettholde workflow'en og reglene som er laget.[29]

Figur 2 The domain i en webapplikasjon

2.2.2 Store

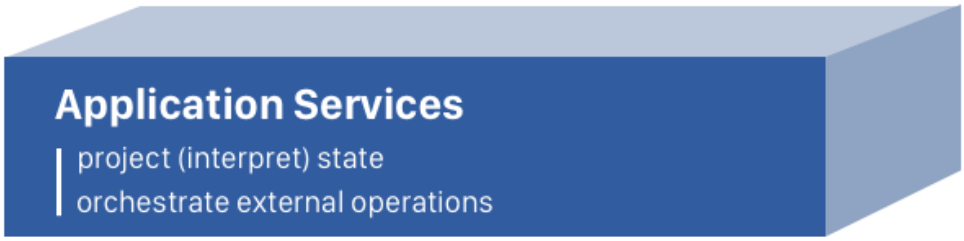
«The Store» i en webapplikasjon er den som holder på staten til applikasjonen og forteller om endringer som blir gjort i staten når de blir gjort. [29]



Figur 3 The store i en webapplikasjon

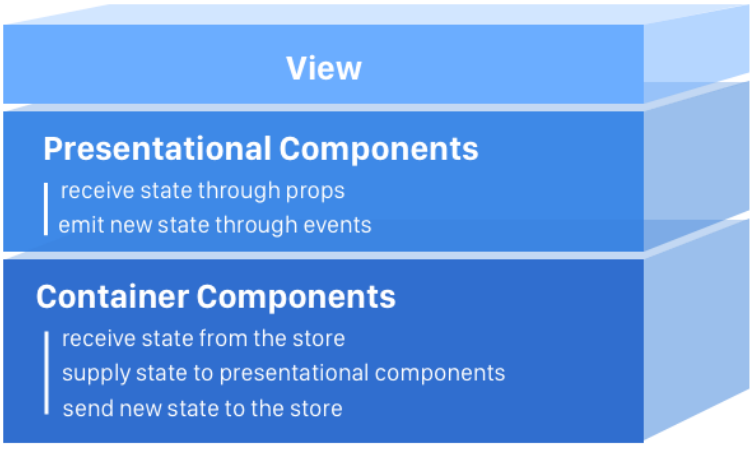
2.2.3 Application service

Application service gjør alt mulig av oppgaver for å holde en hvis flyt når det gjelder endringer i staten. Den kan også gjøre forskjellige oppgaver utenom staten. For eksempel kan den sette stor bokstav på navn i en applikasjon. Dette er ikke en ordinær operasjon, men kan gjøres for å beholde flyten.[29]



2.2.4 The view layer

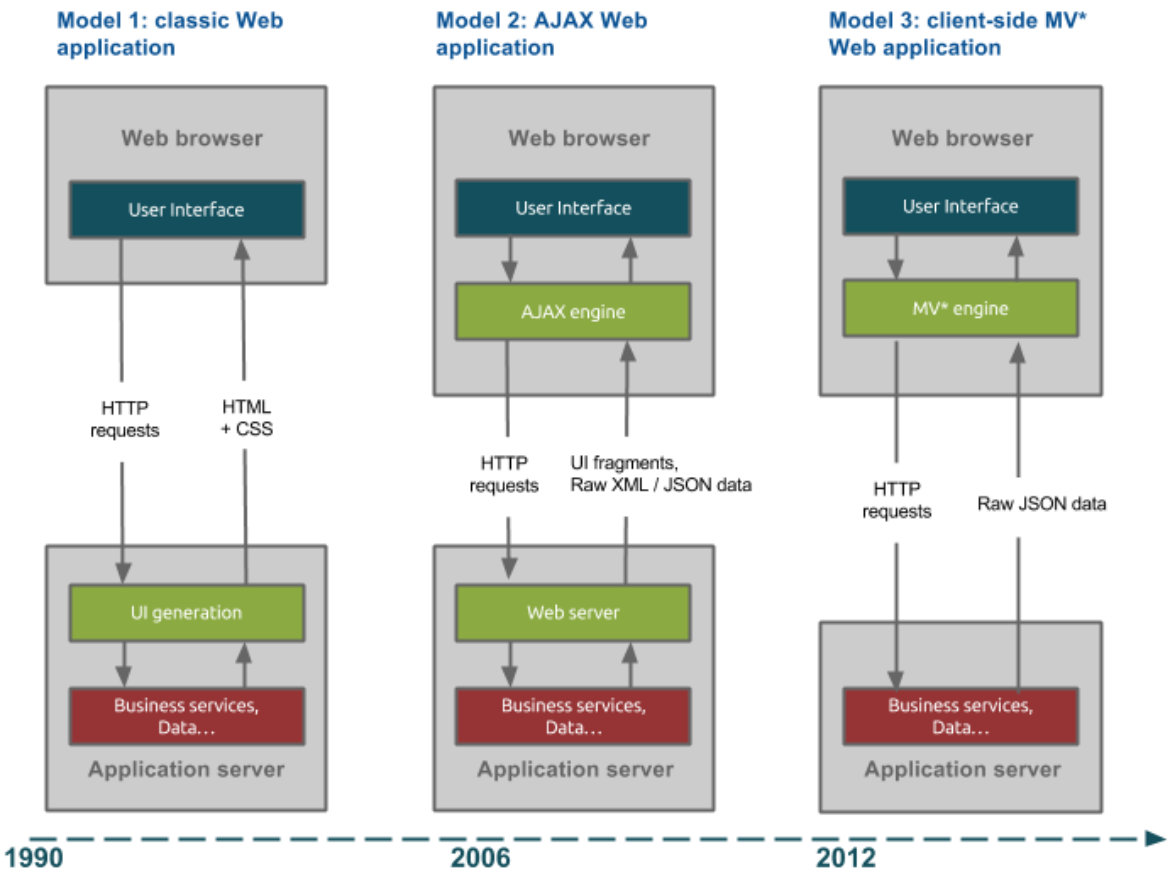
Som figuren under viser, så er dette den største og viktigste delen av oppbyggingen til en webapplikasjon. Det er i denne layeren en bestemmer seg for hvilket rammeverk eller bibliotek en ønsker å bruke for å få applikasjonen slik en ønsker. Her har jeg tatt utgangspunkt i en View layer fra React. View layeren består av to typer komponenter. Den ene er presenterbare komponenter og den andre er container komponenter. Den presenterbare komponenten skal ta mot state gjennom props og lage ny state gjennom forskjellige events. Container komponent skal ta i mot state fra «the store» og gi de videre til de presenterbare komponentene. De skal også sende ny state til «the store»[29]



Figur 4 View layer

2.2.1 Utvikling til Webapplikasjon

En stor endring har skjedd de siste 20 åra innenfor webapplikasjoner sin oppbygging



Figur 5: Web applikasjonens historie[30]

2.2.1.1 Klassisk Web applikasjon

Model 1:

Den første modellen som var mest vanlig før 2000 tallet er basert på at det meste av applikasjonen er server-side og bare enkle CSS og HTML elementer er på klient siden. Her var det ingen tegn til noen avansert Javascript teknologi. [30]

2.2.1.2 Ajax Web application

Model 2:

Den neste modellen er en forbedret versjon av den på 90-tallet, og den kom på midten av 2000-tallet. Nå er det ikke lenger bare helt vanlig HTTP forespørsler som blir sendt, men også AJAX formater som gjør det mulig å hente ut XML og JSON fra server siden. Som et resultat av dette, så var det mulig å få mye mer respons fra web applikasjonen. Ved bruk av Javascript, så skulle applikasjonen kunne bli mye mer dynamisk.[30]

2.2.1.3 Singel-page applikasjon

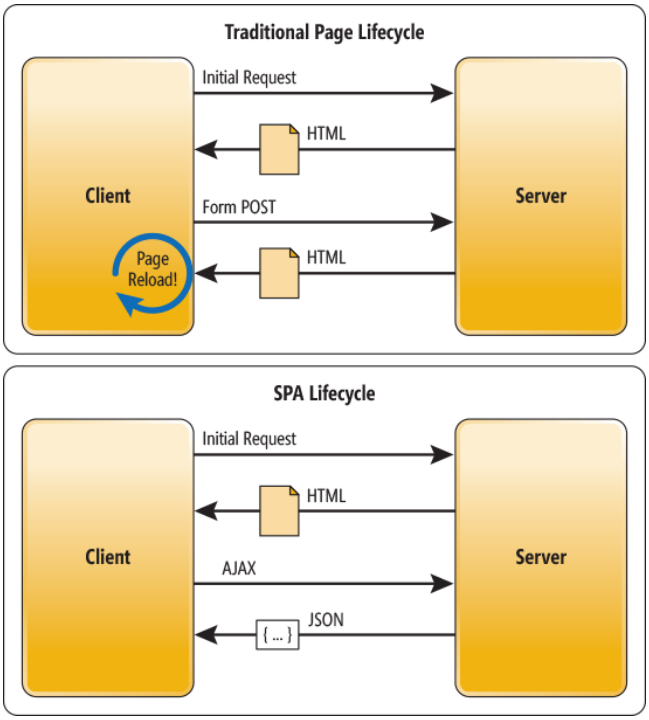
Model 3:

Den tredje og siste modellen er den som er eksisterende i dag og kom tidligst 2012. Denne arkitekturen er unik i forhold til de som har vært tidligere. Her er det en singel-page applikasjon som vil si at en aldri skulle trenge å oppdatere siden. Her har man avansert Javascript som gjør det veldig enkelt for brukeren. Vi ser nå at alt av brukergrensesnitt logikken har blitt flyttet fra server siden til klient siden, der en har forskjellige rammeverk og bibliotek som hjelper deg.[30]

2.3 Singel-page applikasjon

En singel-page applikasjon vil si at applikasjonen oppdaterer seg dynamisk mens brukeren kan ha interaksjoner samtidig som fetching og forskjellige handlinger kan forekomme. Dette er en stor forskjell fra den statiske websiden. Dette har gjort at mye mer av applikasjonen er som tidligere nevnt blitt flyttet til klient siden.

I dag burde du nesten aldri unngå å bruke en SPA løsning på web applikasjonen. Eneste unntaket er om applikasjonen har en helt statisk løsning.[20]



Figur 6 SPA mot tradisjonell struktur[20]

2.4 Kommunikasjon

For å lage en web applikasjon, så må en bruke en eller flere måter å kommunisere på. Det må som nevnt tidligere foregå en dynamisk interaksjon mellom server siden og klient siden. De mest vanlige måtene for en web applikasjon i dag er å bruke en SQL database, REST API og kommunisere gjennom http.

2.4.1 Hypertext transfer protocol

«HyperText Transfer Protocol (HTTP) is an application-layer protocol used primarily on the World Wide Web. HTTP uses a client-server model where the web browser is the client and communicates with the webserver that hosts the website. The browser uses HTTP, which is carried over TCP/IP to communicate to the server and retrieve Web content for the user.

HTTP is a widely used protocol and has been rapidly adopted over the Internet because of its simplicity. It is a stateless and connectionless protocol.» [15]

2.4.2 REST

En fordel med å bruke REST sitt brukergrensesnitt er for å redusere «kobling» mellom systemer. Klient systemene og server systemene kan utvikle seg når den står alene om de holder seg til det definerte brukergrensesnitt. [16]

Noen av de standardene som var før i tiden for web servicer, var veldig store og hadde mye mer funksjonalitet enn det systemene faktisk trengte. Det var også mye overhead når det gjaldt å lage og kjøre forskjellige meldinger. REST var på en måte en svar til alt dette.(32) REST arkitekturen er en mye lettvektet fremgangsmåte og står for Represntational State Transfer. (33) Arkitekturen var utviklet sammen med http protokollen. REST bruker http protokollen for å sende data og bruker URI, status koder som er bygget inn i http.

En REST web service tillater klienter å kjøre «stateless» operasjoner på ressurser som klienten får tilgang til. Når en REST service får en forespørsel, så vil den se på URI'en og metoden som er definert i http forespørselen for å bestemme hva som skal gjøres med forespørselen.

2.4.3 JavaScript Object Notation

JavaScript Object Notation som for det meste blir kalt JSON er et lettvektet data-interchange format som er laget for at det skal være lett for mennesker å skrive og lese. Det er også enkelt for maskiner å sende igjennom og generere.

JSON er byget opp av to strukturer:

- En kolleksjon av navn/verdi par. I de fleste språk er dette ansett som et object, record, struct, dictionary,hash table, keyed list, eller en associative array.
- En ordinær liste av verdier. I de fleste språk er dette antatt som en array, vector, list eller en sequence.[17]

2.4.4 Unified Modeling Language(UML)

UML er definisjonen på en gruppe modeller som skal visualisere hvordan et system er satt opp. Dette er da ofte brukt til å visualisere strukturen i en database og vise hvordan tabellene er satt opp. Det finnes store variasjoner på hvordan en modell kan forekomme, men det er viktig å finne en måte som representerer komponentene dine på en god måte. [41]

2.5 Programmeringsmetodikk

Det er mange muligheter for valg av metodikk når det gjelder å programmere en webapplikasjon. Her vil vi gå inn på hva som er aktuelt å bruke.

2.5.2 Programmeringsspråk

2.5.2.1 Javascript

Javascript er et av de mest populære programmeringsspråkene i verden. Sammen med HTML og CSS, så er Javascript selve grunnsteinene til «World wide web». Javascript er et høynivå programmeringsspråk som ofte brukes til å lage web applikasjoner ved hjelp av forskjellige rammeverk. [9]

2.5.2.2 Typescript

Typescript er en strikt super sett av Javascript som er håndtert av Microsoft. Med et super sett menes det at det er JavaScript som er skrevet og kompilert på en måte som er bedre egnet for enkelte tilfeller. [28]

2.5.2.3 Hypertext Markup Language

Hypertext Markup Language som ofte kalles HTML er et markup language språk som ikke funker særlig bra alene, men funker som regel godt sammen med Javascript og CSS til å lage en web applikasjon. HTML sin oppgave er å organisere dokumentet. HTML er ren tekst, så det er ikke nødvendig å kjøre filen på noen som helst måte.[22]

2.5.2.4 Cascade Style Sheet

CSS eller Cascade Style Sheet som det står for er et språk som beskriver presentasjonen av en webside. Det inkluderer da farger, layout og forskjellige typer fonter som blir brukt. Med bruk av CSS er det mulig å presentere websiden på forskjellige måter.[22]

2.5.3 Rammeverk/Bibliotek

2.5.3.1 React

ReactJS kom i 2013 med Facebook i spissen og er i dag en av de mest stigende Javascript bibliotekene i verden. React ble laget for å kunne håndtere store skaleringer på websider.[12] Det er mange som ser på React som et rammeverk, men i hovedsak er det egentlig et bibliotek. Grunnen til dette er mer fordi React har fokus på fleksibilitet enn at det skal være strenge rammer rundt kodingen.

React egner seg mest for singel-page applikasjoner og mobile applikasjoner med bruk av enten ReactJS eller React native. Med bruk av React sine hjelpemidler skal det være mulig å skape en struktur som gjør det lettere for utvikleren å jobbe, samtidig så vil brukergrensesnittet bli forbedret. Mange store firmaer i dag slik som Netflix, Instagram og ikke minst Facebook.

React har også et vidt åpent bibliotek, der det er mulig å implementere forskjellige verktøy og ting som har blitt laget tidligere. Ved hjelp av slike verktøy som for eksempel Node.js sin NPM(Node Package Manager), vil det være enkelt å implementere en allerede eksisterende løsning i ditt prosj1ekt.

2.5.3.2 Angular

Angular er også et Javascript rammeverk og er i dag håndtert av Google. Angular er et av de mest populære rammeverkene i verden. Angular het før AngularJS og var bestående av bare Javascript, mens dagens versjon er Angular 5 og er programmert med Typescript. Angular har også et stort bibliotek med forskjellige verktøy som hjelper deg å lage applikasjonen på en god måte.[27]

2.5.3.3 Vue

Vue eller VueJS som det også kalles er et progressivt rammeverk som er laget for å bygge gode brukergrensesnitt. Det som gjenkjenner VueJS er at rammeverket har en veldig enkel læringskurve og veldig simpel kodestil. Det er også god dokumentasjon og et veldig nytt og et lettvektet rammeverk.

[26]

Siden det er et lite rammeverk, så gjør det implementasjonen veldig enkel og lett å integrere med forskjellige applikasjoner. Med VueJS kan du bruke både den nyeste Javascript syntaksen ES6 og den litt eldre ES5.[26]

2.6 Design

Designet er det som skal bestemme hvordan web applikasjonen skal se ut og hvor de forskjellige funksjonene skal plasseres. Dette kan virke som en simpel sak, men kan spille en stor rolle.

2.6.2 Wireframe

En Wireframe er en to-dimensjonalt illustrasjon av en nettside sin interface som spesielt har fokus på forskjellige plasseringer av forskjellige elementer, hvilken funksjonalitet og hvordan nettsiden skal oppføre seg.[18]

2.6.2.1 Fordeler med å lage wireframes

Koble sidens struktur sammen med det visuelle designet, ved å vise forskjellige paths/veier.

Gi et klart og tydelig bilde av de forskjellige typene informasjon på brukeren sitt interface.

Bestemme hvilken funksjonalitet som skal være på interfacet.

Prioritere innholdet gjennom å bestemme hvor mye plass å legge av til de forskjellige gjenstandene og hvor de skal plasseres. [18]

2.6.3 Designprinsipper

Designprinsipper kan en se på som en grunnmur i designet. Dette er prinsipper som er basert på psykologiske oppfatninger hos brukeren og kan være veldig troverdige i forhold til riktig utforming av designet.

2.6.3.1 Donald Normans designprinsipper

Donald Norman sine designprinsipper er verdenskjent og har i dag blitt en av de mest omtalte designprinsippene innenfor webdesign. Først handlet egentlig prinsippene hans kun om det psykologiske aspektet, men etter hvert ble det avklart at prinsippene gjaldt også design.

Designprinsippene er inspirert av boken Introduksjon til interaksjonsdesign av Tone Nordbø som har tolket Donald Norman sine prinsipper på en god og strukturert måte.19]

2.6.3.1.1 Synlighet

Synlighet handler rett og slett om å gjøre funksjonaliteten mest mulig synlig for brukeren. Ved et dårlig design, så må en bruker bruke lang tid for å faktisk se funksjonaliteten som det letes etter. Det skal også være lett for brukeren å se hva de forskjellige funksjonene gjør. F.eks. Skal det være lett for brukeren å se om det er en hyperlink som fører deg til en fremmed nettside eller en som fører deg videre inne på den nettsiden du allerede er på.[19]

2.6.3.1.2 Sammenheng

Sammenheng er et prinsipp der det skal være mulig for brukeren å se sammenhengen mellom de forskjellige funksjonalitetene. For eksempel så skal funksjonaliteten til en rad i en tabell ligge på linje og det skal være en klar sammenheng mellom funksjonaliteten og det visuelle en ser.[19]

2.6.3.1.3 Tilbakemelding

En tilbakemelding er alltid viktig for å fortelle brukeren hva som har blitt oppnådd ved å utføre den funksjonaliteten. For eksempel, så er det viktig å fortelle en bruker at en har blitt logget inn, eller at en har blitt registrert i systemet som en ny bruker. Dette er ting som er helt vanlige tilbakemeldinger i en web applikasjon med innlogging og registreringsmuligheter. [19]

2.6.3.1.4 Konsistent design

Konsistent design vil si at det som ser likt ut, skal fungere på samme måte. Det er to forskjellige måter å dele opp konsistent design på: [19]

- **Internt konsistent design**
  Dette omhandler at ting ser ut og oppfører seg tilnærmet likt på tværs av den samme løsningen.

- **Eksternt konsistent design**
  Eksternt konsistent er det som alle i dag tar som en selvfølge. Det at rødt er knyttet til nei og ja er knyttet til grønt er et veldig mye brukt eksempel til dette. Dette er åpenbare konvensjoner som folk i dag er vant til og ikke en god ting å tukle med. Tukling vil bare føre til forvirring hus brukeren, noe som er et tegn på dårlig design.

2.6.3.1.5 Begrensninger

Handler om å gjøre begrensninger, så en unngår at brukeren gjør operasjoner som er feil. En skal alltid tenke at feil som kan gjøres, kommer til å bli gjort og dette må en utvikler være forberedt på og gjøre begrensninger på brukeren sine muligheter til å gjøre minst mulig feil.[19]

2.6.3.1.6 Hint

Hint er et av de viktigste begrepene i et design. Med gode nok hint, så vil en hver bruker forstå de mest viktigste funksjonene i et design som følger gode prinsipper. Et viktig hint kan for eksempel være å skrive syntaksen på det som skal skrives i de forskjellige feltene for en registrering. Eller hint om hvor langt et passord må være og hva det må inneholde for at det skal bli godkjent. [19]

2.6.4 Gestaltlovene

Er basert på gestaltteorien som tar for seg hvordan mennesker tolker helhetlige visuelle inntrykk. Gestaltprinsippene skal gi oss en en helhet om hvordan visuell utforming vil ha på brukerne. Når en bruker gestaltlovene så er det for å øke sjansene for at utformingsprinsippene du har tatt i bruk vil fungere som forventet. [35]

2.6.4.1 Forgrunn og bakgrunn

Mennesker segmenterer geometriske former inn i forgrunn og bakgrunn. Som da kanskje er selvsagt så ligger forgrunnen foran bakgrunnen. [35]

2.6.4.2 Nærhet

Nærhet er når en plasserer elementer sammen for å gruppere de. Dette er et virkemiddel som er ofte brukt i de fleste sammenhengen for å gruppere forskjellige elementer. [35]

2.6.4.3 Likhet

Like elementer hører til i samme grupper. Dette er menneskets oppfatning av ting. F.eks. så kan kvinner, alder, etnisitet være ting som blir gruppert. [35]

2.6.4.4 Sammenkobling

Sammenkobling av forskjellige elementer tilsier at de er relatert på en eller annen måte. [35]

2.6.4.5 Symmetri

Symmetri hjelper oss å sortere geometriske figurer i grafiske brukergrensesnitt. [35]

2.6.4.6 Kontinuitet

Kontinuitetsloven kan bidra til å fremheve struktur i en visuell fremstilling. Elementer som ikke er koblet sammen, men som e i flukt med hverandre ved at de følger usynlige linjer, oppfattes som å høre til en gruppe. Ved å organisere elementer langs usynlige linjer vil observatoren oppfatte disse linjene selv om de egentlig ikke eksisterer. [35]

2.6.4.7 Lukkethet

Loven om lukkethet sier at vi fyller inn manglende informasjon. F.eks. en trekant som ikke egentlig er en hel trekant, men mangler visse trekk for å bli en hel trekant. De manglende linjestykkene fyller hjernen inn for deg og er et trkk som ofte blir gjort i forskjellige logoer. [35]

2.7 Lagring og caching

Lagring og caching er vanlig hos de fleste applikasjoner. Uansett hvilken type applikasjon det er.

2.7.1 Caching

Cache er en term for å lagre noe på kort sikt som også kan brukes igjen om kort tid. Dette vil gjøre at responstiden på å hente det igjen vil bli mye raskere enn den første gangen det ble lastet inn i minnet. Det finnes flere typer cache som er aktuelt for dagens bruk av teknologi. Applikasjon cache, minne cache er de to mest populære. Også Web caching er en viktig faktisk for web baserte løsninger. Hele poenget med en cache i web er å minimere cache trafikken og i tillegg minke responstiden. Caches er i dag funnet på hver level av et systems sin reise fra serveren til browseren. Hele poenget med en cache er å unngå at data må reise helt fra serveren hver gang og heller hente informasjonen fra en plass som ligger nærmere brukeren.

2.7.2 Relasjonsdatabase

"The relational database was invented by Edgar F. Codd at IBM in 1970. The data stored in tables which is made up of columns and rows with data stored inside it. It is then indexed so that it will be easier to find relevant information. Through its lifecycle the data can be read, changed/updated and/or deleted."(31)

I tillegg, så har hver rad en egen unik identitet kalt en primær nøkkel. Disse unike nøklene kan bli linket sammen i forskjellige tabeller og som igjen skaper relasjoner mellom de forskjellige tabellene.

Hver tabell har også en primær nøkkel som kan bli brukt til å definere relasjoner mellom tabeller. Når en tabell rad har kontakt med en rad i en annen tabell, så har den en fremmed nøkkel kolonne som refererer til den primære nøkkelen til den tabellen som den har kontakt med.

I en database tabell, så må hver kolonne ha et navn og en datatype. Navnet beskriver hvilken attributt kolonnen vil ha og datatypen beskriver hvilken type data som skal bli lagret i den. Kolonnen sin type definerer hvilken type verdier som den kan inneholde. (31)

2.7.2.1 Structured Query Language (SQL)

også kjent som SQL er et spørrespråk som er laget for å håndtere data i en relasjonsdatabase. Et SQL uttrykk er et utrykk som er laget slik at det er enkelt å lese, forstå, skape, oppdatere eller slette forskjellig data i en tabell.

2.7.2.2 PostgreSQL

PostgreSQL er et objekt relasjonelt databasesystem som ble utviklet først tidlig på 1980-tallet, men som i dag er håndtert av open-source som vil si at det er gratis og alle har tilgang. [24]

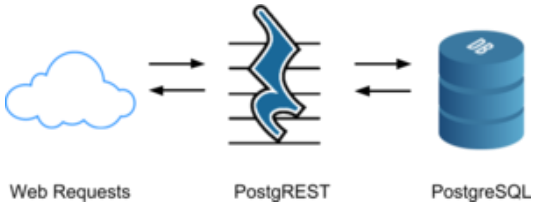
2.7.2.2.1 Views

En View i en database er et object av en query. En kan få tilgang til et view som en virtuell tabell inne i PostgreSQL databasen. Med andre ord, så representerer et view en logisk tabell med forskjell data som er håndplukket gjennom bruk av SELECT metoden. Views sin data blir ikke fysisk lagret, så det er ikke mulig å gjøre noe annet enn å hente og lese dataen.[25]

- En view kan for eksempel hjelpe til med å forenkle en avansert query ved å utføre en enkel SELECT kommando[25]
- Ved å lage et View, så kan du hjelpe brukeren å se akkurat den dataen som ønsket og droppe alt det andre, eller droppe informasjon som kanskje ikke brukeren har lov til å se.[25]
- En View gir en god representasjon av dataen uansett om dataen som ligger under endrer seg, så vil view'et oppdatere seg og endres. [25]

2.8 PostgREST

PostgREST er en hurtig og enkel løsning og implementasjon som et slags lag ovenfor databasen. Når dette laget er implementert, så vil det automatisk lage API'er av tabellene dine, som igjen videre kan bli sendt og mottatt av web klienten.



3. Materialer og metoder

I denne seksjonen vil jeg se på hvilke metoder, materialer, bibliotek og rammeverk som jeg valgte å bruke for denne oppgaven.

3.1 Metoder

En gjennomgang av hvilke metoder som jeg valgte å bruke under programmeringsperioden i bacheloroppgaven. Jeg var alene om denne oppgaven, så metodene ble tilpasset til det forbruket jeg hadde.

3.1.1 Prosjekt planlegging

Prosjekt planlegging er den delen hvor en må ta et valg ang. utviklingsprosessen. Det er her en velger hvilken sti en vil gå. Om en vil ha en bratt bakke, en trall eller hvilken prosjektmodell som måtte passe prosjektet best.

3.1.1.1 Wireframes

En Wireframe er en to-dimensjonalt illustrasjon av en nettside sin interface som spesielt har fokus på forskjellige plasseringer av forskjellige elementer, hvilken funksjonalitet og hvordan nettsiden skal oppføre seg.

3.1.1.1.1 Fordeler med å lage wireframes

- Koble sidens struktur sammen med det visuelle designet, ved å vise forskjellige paths/veier.

- Gi et klart og tydelig bilde av de forskjellige typene informasjon på brukeren sitt interface.
- Bestemme hvilken funksjonalitet som skal være på interfacet.
- Prioritere innholdet gjennom å bestemme hvor mye plass å legge av til de forskjellige gjenstandene og hvor de skal plasseres.
- Wireframes er en metodikk som gjør det enkelt for både kunde og utvikler å få et overblikk over hvordan designet og strukturen på en web basert nettside skal være. [18]

3.2 Materialer

Materialer er alle verktøy som har blitt brukt gjennom prosjektet og vært nødvendig for at prosjektet ble gjennomført på en god måte.

3.2.1 Microsoft Visual Studio

Microsoft Visual Studio er en IDE laget av Microsoft. Den brukes til å utvikle dataprogrammer, web applikasjoner, mobil applikasjoner og web services. Microsoft som også står for Windows, og bruker derfor windows API, Windows form og andre Windows funksjoner. Visual Studio støtter opp til 36 forskjellige programmeringsspråk og har et veldig fint system for debugging. Den har også integrert Git, slik at det er enkelt å koble prosjektet sitt opp mot en Git, og gjøre alt som trengs via IDE'en.(11)

3.2.2 Postman

Postman er en google Chrome applikasjon som skaper en interaksjon med http API'er. Det gir deg en god GUI som lager forskjellige forespørsler og lese spørringer med JSON eller XML. Dette kan brukes til å teste ut applikasjonen din og finne ut hvordan den reagerer til forskjellige spørringer.(1)

3.2.3 Swagger

Swagger er et open source software rammeverk som gir store mengder av verktøy som kan hjelpe deg å designe, bygge dokumentere og hente ut data med RESTful web services. (2)

3.2.4 PgAdmin

pgAdmin er en av de mest populære open-source administrasjon og utviklings platform for postgreSQL. Det er den mest avanserte database i verden. Der kan du lett håndtere alt som trengs å gjøres med en database. Veldig enkelt og grei platform å jobbe med.(3)

3.2.5 React developer tool

"Adds React debugging tools to the Chrome Developer Tools.

React Developer Tools is a Chrome DevTools extension for the open-source React JavaScript library. It allows you to inspect the React component hierarchies in the Chrome Developer Tools.

You will get a new tab called React in your Chrome DevTools. This shows you the root React components that were rendered on the page, as well as the subcomponents that they ended up rendering."(4)

3.2.6 SonicWall VPN Client

VPN gjør det mulig å være tilkoblet til en IP adresse som er langt unna deg og gjøre dine behov selv om du ikke har muligheten til å være fysisk tilstede. SonicWall sin

VPN client er enkel og grei for alt en VPN skal brukes til, så lenge en har riktig informasjon slik som passord, brukernavn og riktig IP tilgjengelig.(5)

3.2.7 Jira

Jira er et system av Atlassian, som er brukt for å holde kontroll på forskjellige problemer eller gjøremål i et prosjekt. Jira er det mest brukte verktøyet innenfor agile metoder. Jira gjør det mulig å følge hele livssyklusen til et problem hele veien til den er ferdig løst.(6)

3.2.8 Confluence

Confluence er mest brukt til prosedyre. Confluence er en wiki hjelpemiddel som hjelper teams å samarbeide og dele kunnskap. Ved hjelp av Confluence er det mulig å gi oppgaver til spesifikke brukere, samt håndtere flere kalendere samtidig. Veldig til hjelp i flere teams av utviklere rundt om for å holde strukturen i et prosjekt.(7)

3.2.9 Github

Github er en web basert hosting service for versjon kontroll med bruk av Git. Github gir deg mulighet til å ha kontroll på koden din på flere forskjellige måter. Det er mulig å søke et feilene sine i Github, se hva som har blitt gjort til en hver tid og samtidig ha en wiki side for hver prosjekt som er laget.(8)

3.2.10 Git

Git er et versjon kontroll system, som holder styring på koden, slik at det skal være mulig å gå tilbake til en tidligere utgave om en endring ikke skulle virke. Dette er noe som er veldig relevant for alle som programmerer i dag.

Git brukes ofte sammen med en hosting service slik som Github. (10)

3.2.11 JsonView

Er en Chrome extension som rydder opp en json fil i nettleseren din. Gjør Json format mye lettere å lese og skaper en god og fin struktur på innholdet.(14)

3.2.12 postgREST

"PostgREST is a standalone web server that turns your PostgreSQL database directly into a RESTful API. The structural constraints and permissions in the database determine the API endpoints and operations."[21]

Ved hjelp av PostgREST kan vi bruke disse endpoint'ene til å enklere å håndtere og få tilgang til dataen gjennom API'et.

3.2.13 dbDiagram

dbDiagram er et gratis UML diagram der du enkelt kan sette opp strukturen som du har i databasen din. Du har flere forskjellige verktøy som kan eksportere til database syntaks, PDF eller bare et enkelt bilde. [34]

3.2.14 Word

3.3 Rammeverk og bibliotek

En liste over de forskjellige bibliotekene og rammeverk jeg har valgt å bruke.

3.3.1 Rammeverk

For å lage en web applikasjon i dag, så er det viktig å velge de riktige alternativene, som gir deg den web applikasjonen du vil ha. Et rammeverk har stor betydning på hvordan syntaksen i en web applikasjon er, og hvordan applikasjonen blir bygd opp.

3.3.1.1 ReactJS

React sitt bibliotek er ikke et stort et, men det er den lette håndteringen av biblioteket, samtidig som en god håndtering av DOM som skal være grunnen til den stigende populariteten til Javascript biblioteket React.

3.3.2 Node.js

Node.js er en platform som er bygget på JavaScript runtime. Node.Js er laget for å kunne lett lage webapplikasjoner som også er justerbare. Node.js sin NPM gjør det enkelt å implementere forskjellig kode i din applikasjon. Node.JS ble opprettet av Ryan Dahl i 2009 og er i dag brukt av en rekke store firmaer som blant annet Yahoo!, Microsoft og Paypal.(13)

Ved hjelp av Node package manager, så har jeg hentet ut en del verktøy som har hjelpet meg å få en god web applikasjon. Installasjonen er så enkel som at en først installerer NPM ved å laste ned og installerer ved hjelp av kommando vinduet. Deretter er det bare å installere de forskjellige pakkene i det prosjektet en skulle ønske å bruke det i.

De forskjellige pakkene jeg har brukt i prosjektet er:

@React-table
ble lagt til å prosjektet for å få en ordentlig og ryddig tabell[39]

@React-bootstrap

For å implementere en god og ordentlig nav meny og forskjellige ikoner.[40]

@React-router-bootstrap

For å kunne linke de forskjellige sidene på en fin måte.[37]

@Nivo/pie

Implementere en fin og strukturert pie chart[38]

4. Resultater

Web applikasjonen som har blitt utviklet har fokus på bruk av det nyeste og mest populære rammeverket innen for singel-page applikasjoner. Applikasjonen er koblet til en back-end og de kommuniserer med hverandre gjennom PostgREST. Databasen som ligger i bunn er en postgreSQL database som ligger hos Cordel. Databasen henter ut reell data fra Cordel sine systemer. På Client-siden er det brukt React sitt bibliotek for å få frem singel-page applikasjonen.

4.1 Webapplikasjon arkitektur

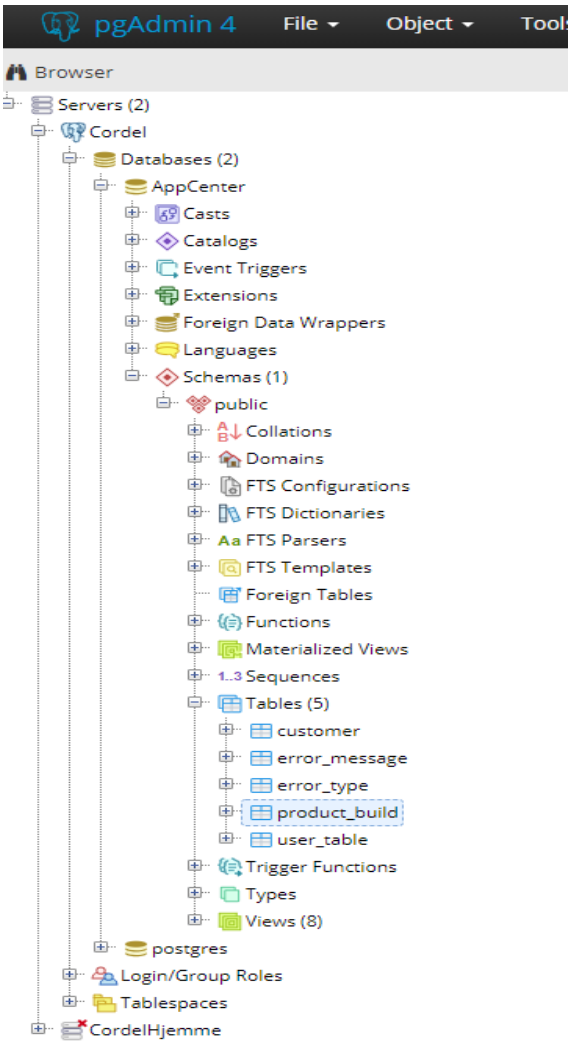
Figur 7 en simpel arkitektur av webapplikasjonen

Figur 5 viser en enkel og oversiktlig figur av strukturen til web applikasjonen jeg har laget. Hver boks tilsier en level og hele poenget er å vise hvordan applikasjonen er

knyttet sammen og hva som ligger hvor i forhold til nivåene. Vil gå nærmere inn på detaljene og forhåpentligvis gi en klarere visualisering av prosjektet og webapplikasjonens arkitektur.

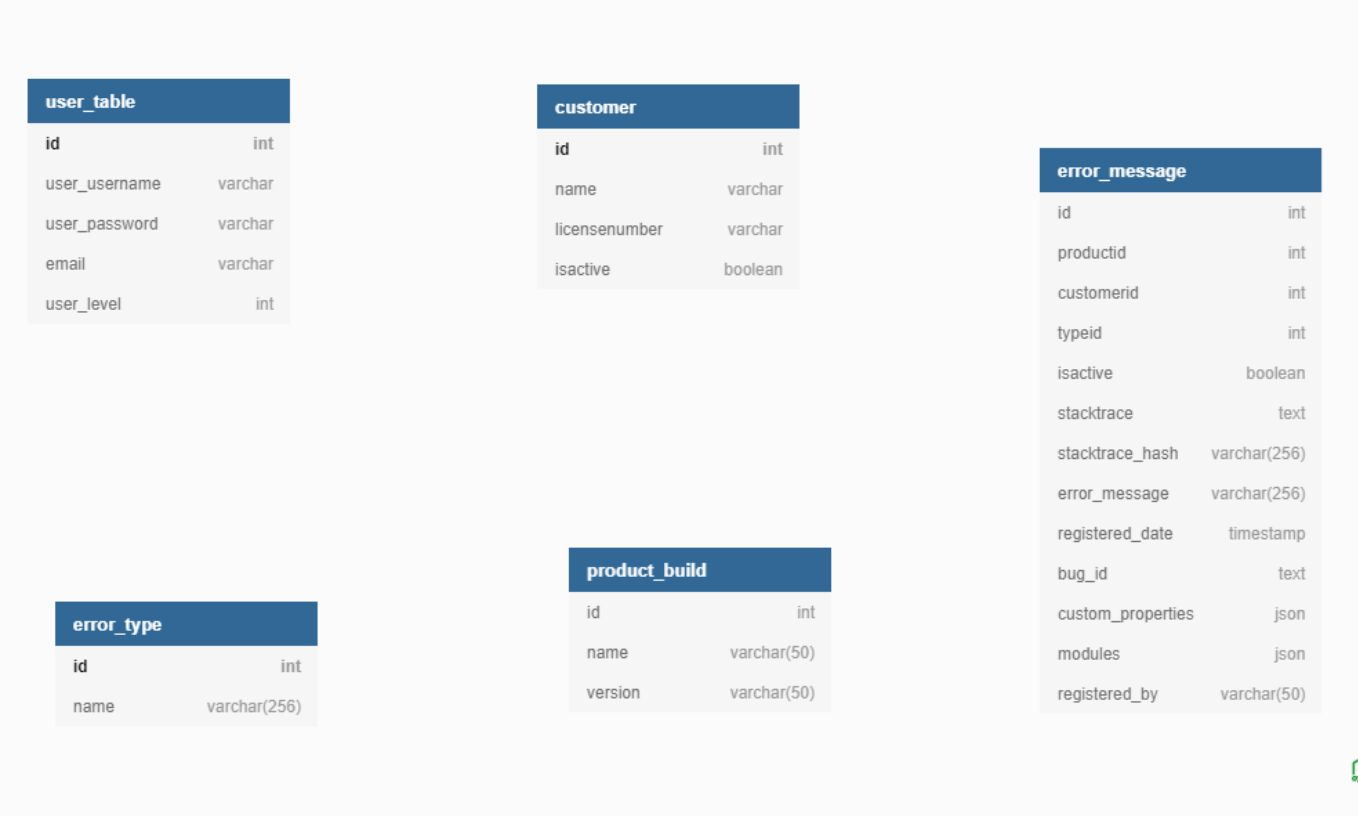
4.2 Database

Databasen er det Cordel som har stått for og er en postgreSQL database. De har gitt meg tabeller som jeg kan jobbe med og lage til forskjellige Views med den dataen som ligger i de forskjellige tabellene.



Figur 8 Viser strukturen til databasen vist i pg4 Admin

4.2.1 PostgreSQL tabeller



Figur 9 Database tabellene som er brukt i applikasjonen

4.2.1.1 Customer

Customer tabellen representerer en kunde i systemet til Cordel og inneholder:

- Id
- Navn
- Lisens nummeret til kunden
- IsActive som vil fortelle om kunden bruker aktivt systemet i dag.

4.2.1.2 Error_message

Presenterer en feilmelding med en egen id for feilmeldingen, produktet, kunden og type og en boolean IsActive, der den er aktiv om feilmeldingen ikke er håndtert. Den vil også inneholde en stacktrace som gjør det mulig for utviklerene til Cordel å se kodefeilmeldingen som oppstår. Det er også lagt ved alt annet av relevant informasjon som trengs for å løse problemet.

- Id
- ProduktId
- CustomerId
- TypeId
- IsActive
- Stacktrace
- Stacktrace_hash
- Error_message
- Registered_date
- Bug_id
- Customer_properties
- Modules
- Registered_by

4.2.1.3 Error_type

Presenterer hvilke forskjellige typer feilmeldinger som finnes i systemet

- Id
- Name

4.2.1.4 Product_build

- Id
- Name
- Version

Product_build vil si hvilken applikasjon feilmeldingen har oppstått på. Så det skal være mulig å se hvilken versjon som feilmeldingen har oppstått på og da navnet og id'en på versjonen.

4.2.1.5 User_table

- Id
- Use_username
- User_password
- Email
- User_level

Gjør det mulig å opprette en bruker med eget brukernavn, passord, mail og definere en level for hvor mye brukeren vil ha rettigheter til å gjøre. F.eks. så vil en admin ha muligheten til å slette brukere eller opprette nye brukere. Dette er en tabell som Cordel mest sannsynlig vil brukere i fremtiden, men ikke var i fokus når jeg laget webapplikasjonen.

4.2.2 Views

Som nevnt i teori delen tidligere, så er Views en fin funksjon for å hente tak i den informasjonen som er nyttig. For å få ut alt av informasjon som er nødvendig for å lage webapplikasjonen etter spesifikasjonene og ønskene til Cordel, så har vi sammen kommet frem til åtte forskjellige Views. Hvert View representerte data som skulle hjelpe Cordel på veien mot å effektivisere feilmeldingssøkingen. Views ble enkelt opprettet og håndtert gjennom bruk av open-source programmet pgAdmin 4. Med bruk av pgAdmin 4 er det innebygget eget query tool og et enkelt design som har de hjelpemidlene som trengs for å håndtere en PostgreSQL database på alle mulige måter.

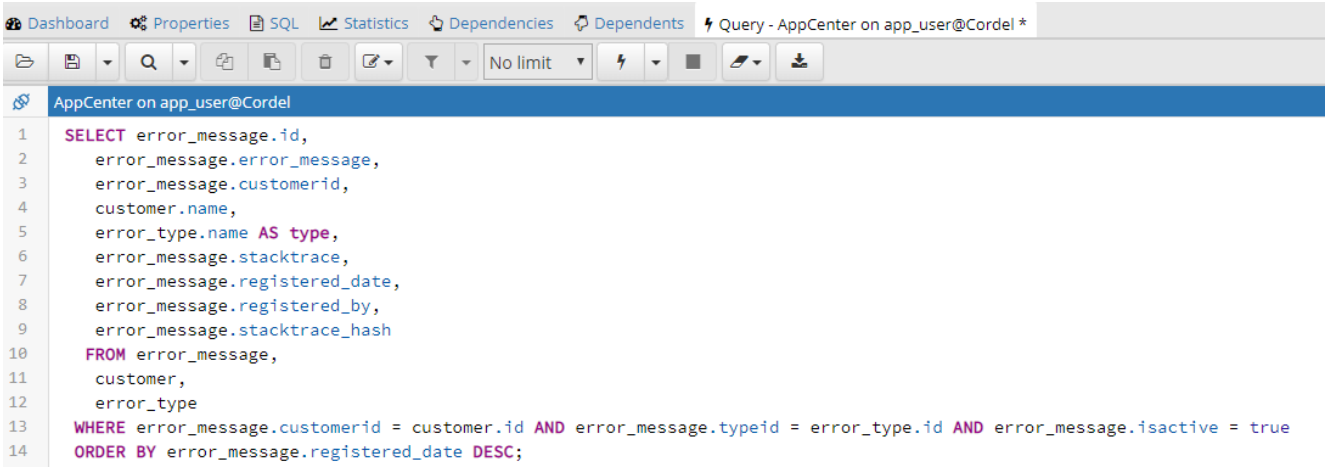
Ved bruk av VPN tilgangen som Cordel hadde gitt meg, så kunne jeg enkelt håndtere databasen hjemmefra ved bruk av pgAdmin4 og ved innlogging i databasen.

4.2.2.1 View struktur

Eksempel figuren under viser en av de åtte view'ene som ble laget til dette prosjektet. Dette ble laget ved bruk av informasjon av de fire fysiske tabellene som ligger i databasen og ble laget for å vise en oversikt hver enkelt feilmelding som kommer inn.

- En SELECT er brukt først for å velge hvilke verdier som skal hentet ut fra de eksisterende tabellene.
- AS type blir brukt for å definere
- FROM blir brukt for å definere hvilke tabeller dataen blir hentet fra.
- WHERE blir ofte brukt for å avklare kriterier, slik som i eksempelet, så blir det avklart at customerid og cusomer.id er de samme verdiene.

- AND blir brukt for å slippe å duplisere kode. Så en kan heller bruke AND i stedet for å måtte bruke WHERE flere ganger. Dette er da også bedre kodestil.
- ORDER BY er for å skape sortering og i dette tilfellet bruker en sortering etter datoen feilmeldingen kom inn i databasen. Og DESC sier at datoen er desimaler.



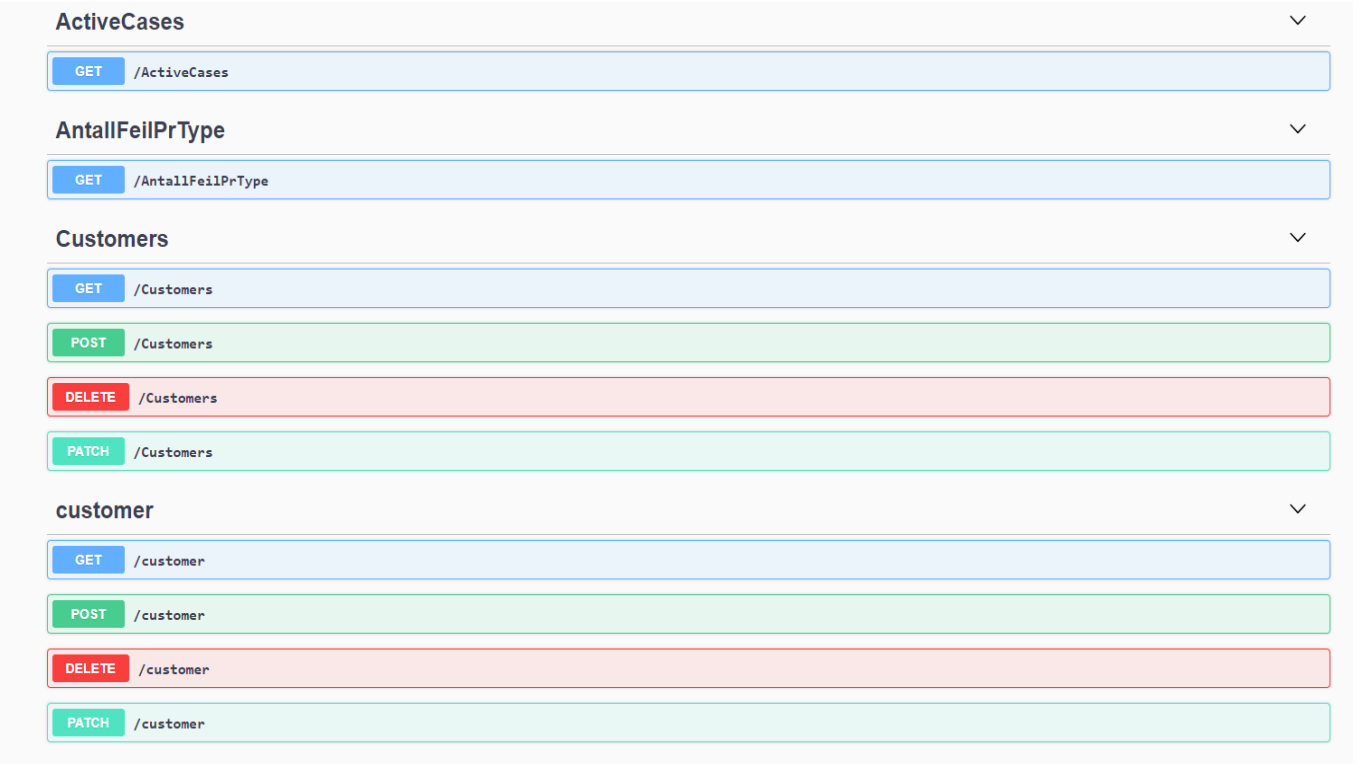
Figur 10 View strukturen i en PostgreSQL database

4.3 PostgREST

Her går jeg mer inn i detaljer hvordan jeg brukte PostgREST som en del av web applikasjonen. Som tidligere nevnt i 2.6, så er PostgREST et populært valg innen for bygging av applikasjoner. På figuren under ser vi et bilde av strukturen til PostgREST som et mellomledd mellom Web og databasen. For hver gang vi laget nye virtuelle tabeller, så ble disse automatisk oppdatert og fikk endepunkter på sekundet som enkelt kunne bli implementert i applikasjonen.

4.3.1 Swagger

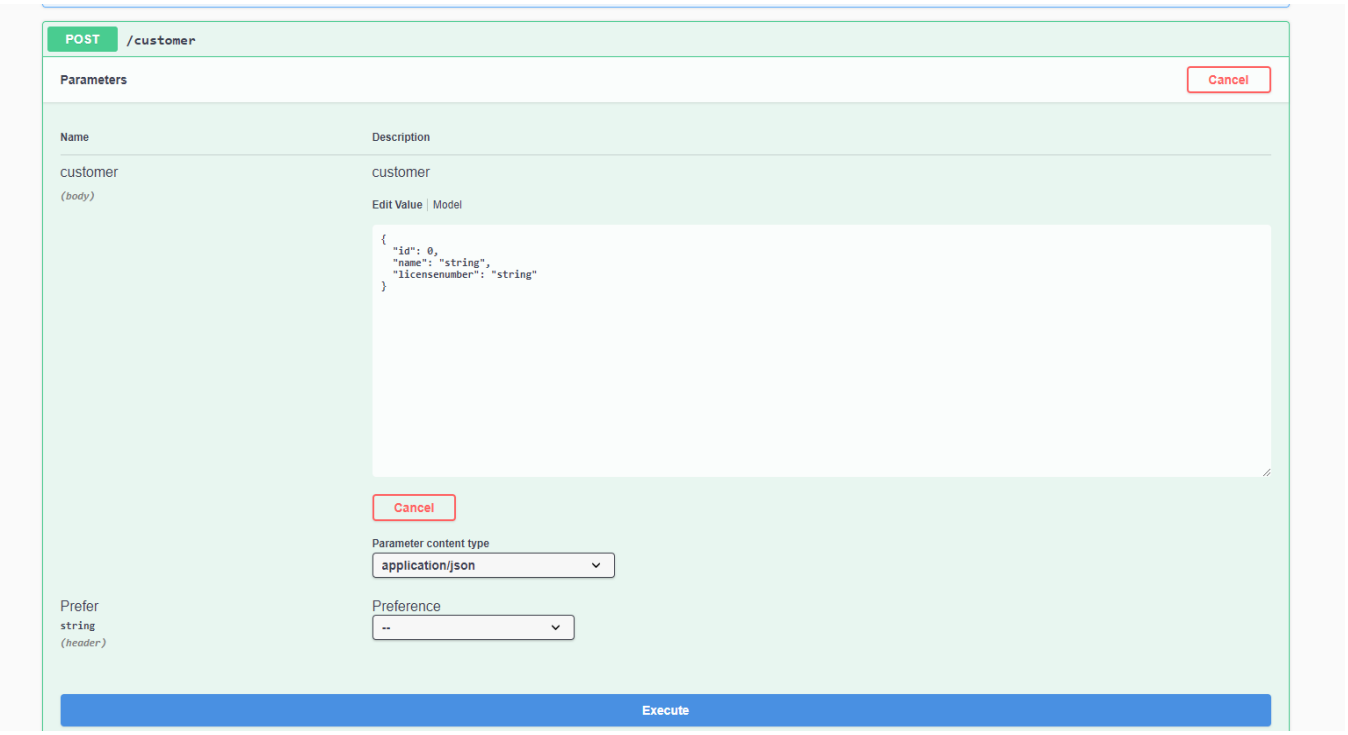
Swagger som tidligere nevnt(Kapitel 3.14) er et hjelpemiddel som passer utmerket til bruk av PostgREST fordi Swagger UI er en fin implementasjon som gir en god struktur over alt av data PostgREST gir.



Figur 11 Swagger UI

Her kommer alt av data som endepunkter og enkelt å håndtere i hvilken som helst klient side som en måtte ønske å bruke. Ser også fra UI'en at det bare er mulig å hente dataen fra de forskjellige View'ene som ble laget, mens en kan gjøre flere forskjellige CRUD operasjoner på de tabellene som er fysisk lagret i databasen.

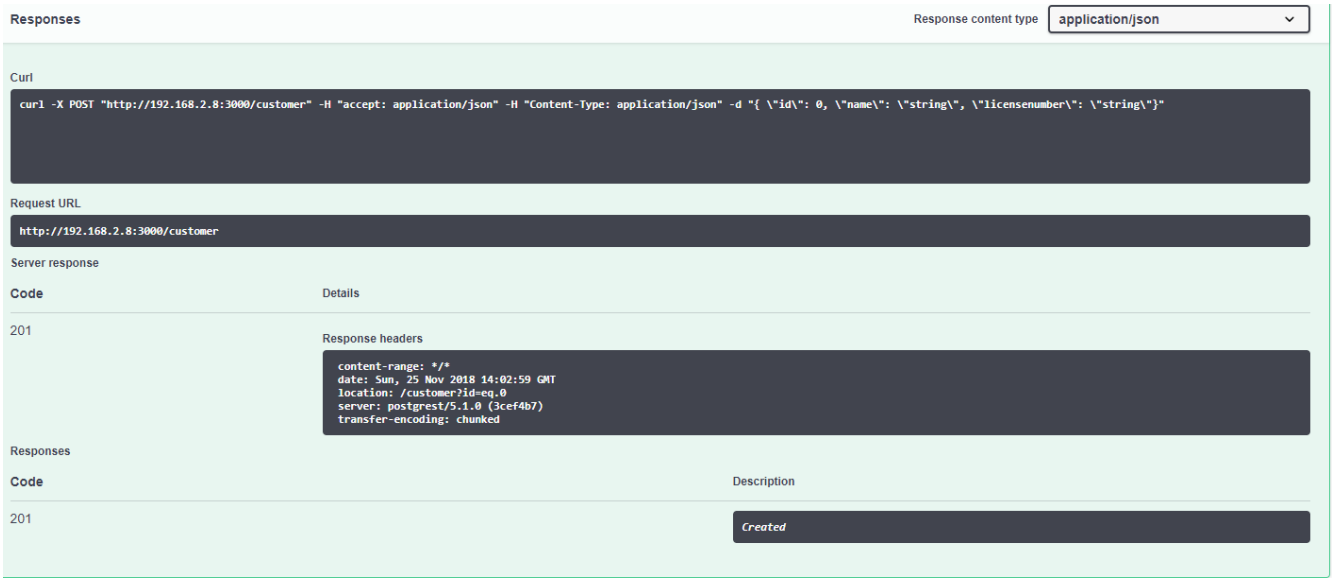
4.3.1.1 Utnytting av Swagger sin verktøykasse



Figur 12 Swagger operasjoner

Som vi ser på figur 10, så er det også mulig å sende data direkte fra Swagger. Dette er også en fin funksjonalitet som kan hjelpe deg å teste ut forskjellige muligheter med APi'et. Der du kan fylle ut modellen og få kjørt data rett inn i databasen på samme måte som en ville gjort det ved en web applikasjon.

4.3.1.2 Respons



Figur 13 Swagger respons

Ved å trykke på «Execute» så vil også Swagger Ui'en gi deg en respons der en får URL'en til der siden hvor dataen haner, samt en CURL syntaks om dette skulle være til hjelp. I hovedsak ble denne brukt til å få URL'en til dataen som inneholder JSON-format med alt av data som er på de forskjellige tabellene som blir kjørt, både View og de fysiske tabellene har sin egen vei. For eksempel så vil hele kundetabellen til Cordel ha denne syntaksen URL:PORT/customer.

4.3.1.3 JsonView

Syntaksen til JSON-filen som vil bli vist vil ikke ha et format som er noen form for ryddig, så brukte jeg JsonView. (som er forklart i 3.2.14.)

4.4 Klient-side

Det er veldig mange valg når det gjelder oppbygging av en klient side til en web basert applikasjon. Her vil jeg gå mer inn i detaljer om hva som har blitt brukt for å frem applikasjonen sin klient-side.

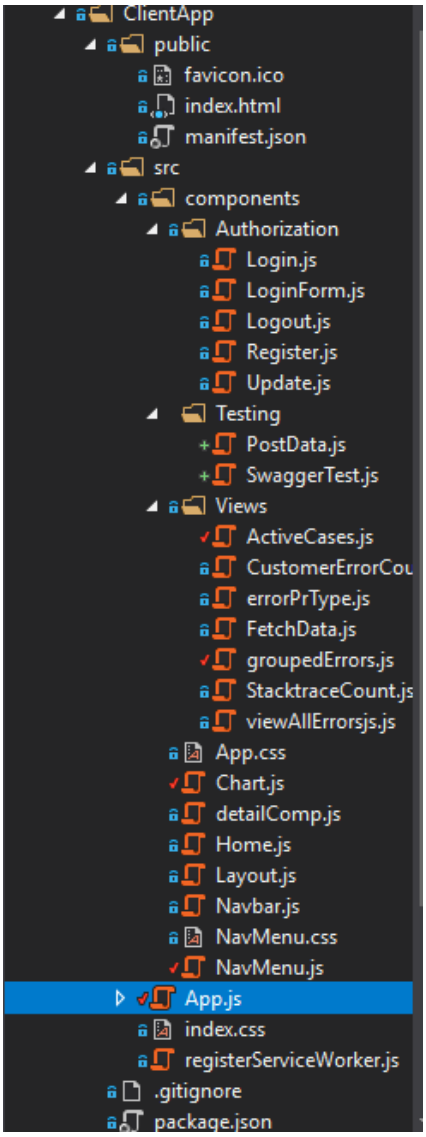
4.4.1 Programmeringsspråk

I valget av programmeringsspråk så var det ganske åpenbart at Javascript var det som var mest aktuelt siden det var en web basert applikasjon som var målet med oppgaven. Typescript var også aktuelt, men endte til slutt opp med å bruke Javascript. Siden React biblioteket gjør det mulig å bruke både JavaScript og TypeScript, så hadde begge vært optimale valg.

4.4.2 Kode struktur

React som bibliotek skaper en stor fleksibilitet hos applikasjonen. Det er veldig mange måter du kan velge å utvikle applikasjonen på.

4.4.2.1 Mappestruktur



Figur 14 Mappestruktur i applikasjonen sin klient side

Figur 14 viser hvordan jeg har valgt og strukturert prosjektet i min oppbygging av klient siden til prosjektet.

- Views

  Er en samling av alle tabell komponentene, der de fleste er relativ like, med mye av de samme funksjonalitetene, med da selvfølgelig forskjellig typer data.
- Testing

  Inneholder forskjellig test filer til evt. Videre utvikling av applikasjonen.
- Authorization

  Inneholder alt som har med autorisering å gjøre.
- Annet

På annet er det app.js som er selve rattet i applikasjonen. Er der alle komponentene blir satt i gang.

4.4.2.2 Kodestruktur

I React så er det lagt fokus på å splitte det meste av brukergrensesnittet inn i forskjellige komponenter. Dette da for at det skal være mulig å bruke opp igjen de forskjellige komponentene.

Nå vil jeg ta for meg de vanligste kodesnuttene jeg har brukt i denne web applikasjonen og hva de gjør.

4.4.2.2.1 Constructor

I alle mine komponenter så starter jeg alltid med å lage en konstruktør eller da en constructor på engelsk. Dette blir gjort fort å definere props'ene med super(props) og for å sette en state. På Figur 15 som er et eksempel på en konstruktør, som er tatt fra et av komponentene jeg laget. Der her ser vi at customersOnType blir definert og som skal da etter en konstruktør bli brukt videre i en componentDidMount metode.

```
constructor(props) {
    super(props)
    this.state = {
        customersOnType: [], row: {}
    }
```

Figur 15 constructor

4.4.2.2.2 componentDidMount

Her er et eksempel på componentDidMount fra applikasjonen. Den blir kalt for det er en god plass å gjøre et nettverkskall. Ser på figuren under at et nettverkskall blir gjort, som venter på en JSON tilbake. En setState blir også kalt . Dette blir gjort for å trigge en ekstra rendering for å sikre at brukeren får sett riktig state.

Ser også her at jeg valgte å bruke fetch som er vanilla javascript. Dette gjorde jeg for å slippe å bruke et ekstern bibliotek for å håndtere http forspørslene gjennom react web applikasjonen. Det er flere bibliotek som tilbyr akkurat denne funksjonen, men ser ingen nødvendighet i å bruke det når det ikke er noen stor forskjell fra fetch.

```
async componentDidMount() {
    let response = await fetch(this.url + "/errorTypeOnCustomer?stacktrace_hash=eq." + encodeURIComponent(this.props.stacktrace_hash))
    let json = await response.json()
    this.setState({ customersOnType: json, loading: false });
}
```

Figur 16 componentDidMount

4.4.2.2.3 Importering

Som tidligere nevnt i rapporten, så er en av fordelene til React at det er så mange som bruker biblioteket og det er derfor et veldig stort utvalg og muligheter å implementere. En trenger ikke alltids å finne opp kruttet på nytt og det er flere funksjoner som er bedre enn det du kunne ha laget selv.

Med bruk av Node.js sin package manager (NPM), så har jeg implementert flere bibliotek som passer til react. Bibliotekene blir også godt dokumentert på brukermetoder, så implementasjonen er relativt enkelt.

Er bare å laste ned NPM også finner en installeringsmetoder sammen med bibliotekene når du finner de du ønsker å bruke. F.eks. en måte er;

«Npm install nivo –save»

Dette vil da installere hele biblioteket til nivo, noe som er ganske stort, så derfor ville vi kanskje ha installert mer spesifikt. Slik som nivo/pie som gir oss et pie chart (som er beskrevet i kap.5.2.5)

På Figur 17 ser vi måten å importere forskjellige bibliotek etter at de er installert gjennom bruk av NPM.

```
1  import React, { Component } from 'react';
2  import { ResponsivePie } from '@nivo/pie'
```

Figur 17 Importering av bibliotek

På Figur 18 ser vi hvordan bruken av bibilioteket/metoden er i React. Slik blir det også gjort om en skal implementere et annet komponent. Syntaksen på et bibliotek med

forskjellige verdiene som en kan bruke er som regel lagt med eller linket til i en ekstern webside.

```
<ResponsivePie
    data={this.state.error_types}
    margin={{
        "top": 40,
        "right": 80,
        "bottom": 80,
        "left": 80
    }}
    innerRadius={0.5}
    padAngle={0.7}
```

Figur 18 Bruk av importerte metoder/bibliotek

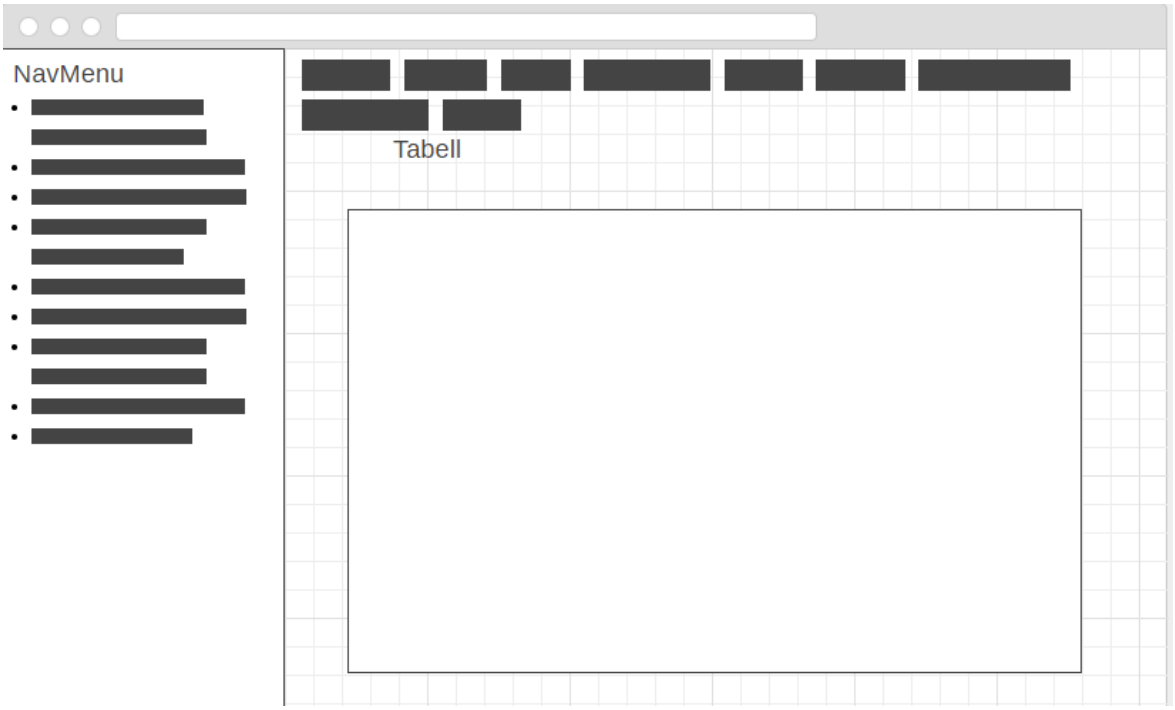
4.5 Design

Før en i heletatt starter med å implementere noen form for design på applikasjonen, så burde det alltid bli laget wireframes. Mine wireframes ble først tegnet for hånd sammen med Cordel og deretter implementerte jeg de inn i et wirefram program. Å tegne noe for hånd er et ok utgangspunkt om du har tegnet selv, men ved å bruke wireframes ble det kjapt en god del mer profesjonell skisse over designet til applikasjonen.

4.5.1 Wireframes

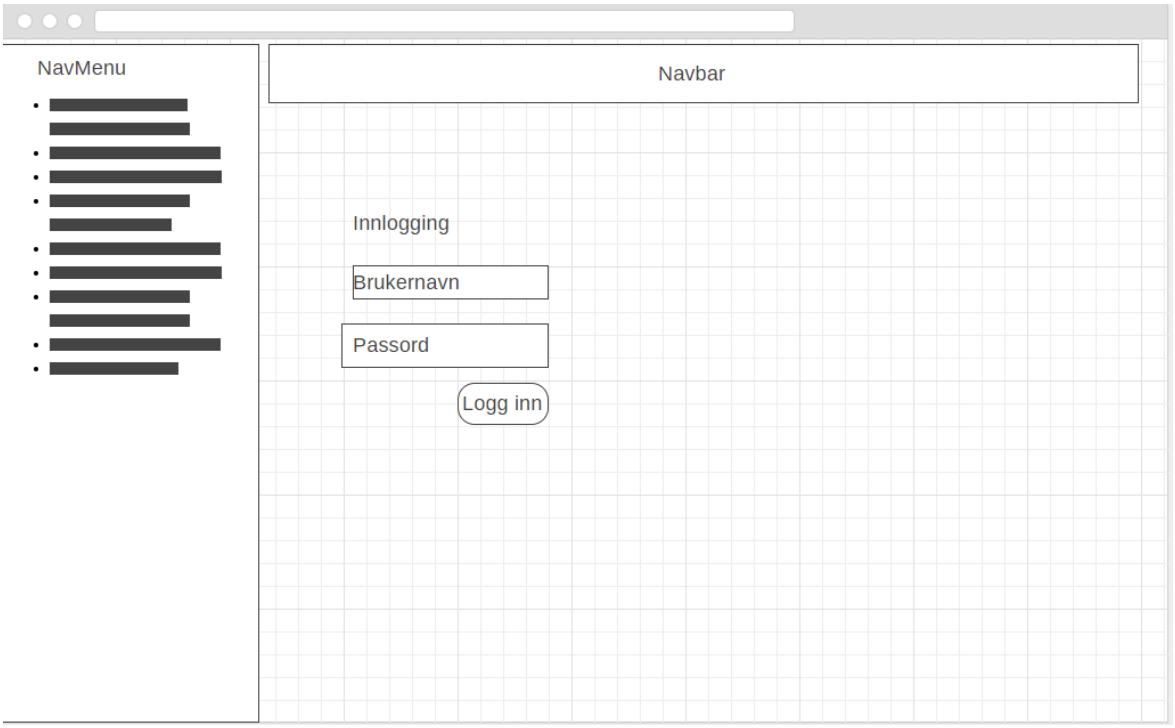
Wireframes sin viktige rolle i designet til applikasjon pratet jeg om i kapitel 3.1 og dette er noe jeg valgte å ta i bruk nettopp med hensyn til hva bedriften tenker ang. designet.

I første omgang var det tre wireframes som var mest aktuelle for Cordel at jeg implementerte:



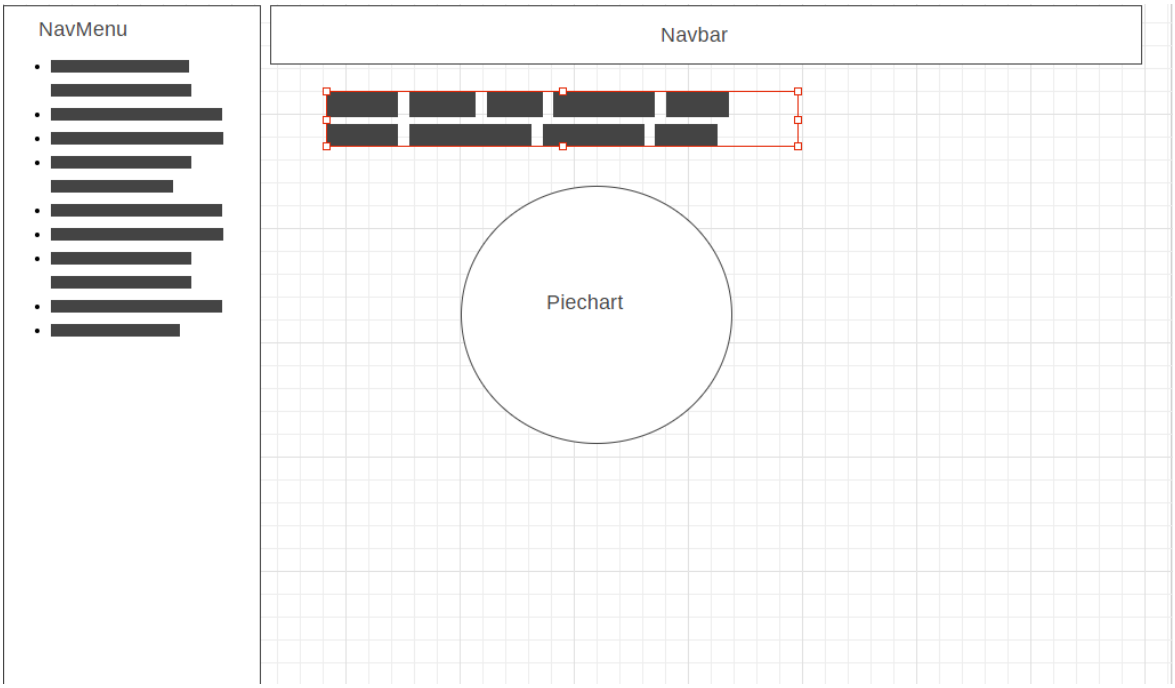
Figur 19 Wireframe av tabell sidene sin struktur

Dette er wireframe strukturen som vi bli enige om før noen form for programmering ble påbegynt. En NavMenu på siden som linker til de forskjellige nettsidene og en tabell som skal vise de aktuelle dataen. En veldig simpel wireframe, men som både jeg som utvikler og bedriften kunne si oss fornøyd med.



Figur 20 Wireframe av innloggingen

Innloggingen er også relativt simpel, der en ikke får en mer standard innlogging enn dette. Grunnen til dette er mer for at autorisering ikke skulle være i hovedfokus for dette prosjektet.



Figur 21 Wireframe av GUI'en til dashboardet

Samme som de to andre to wireframes'ene, så er denne også simpel. Ble enig med bedriften om å holde et så stilrent design som mulig. Denne figuren viser da dashboardet til applikasjonen, der det skal være en piechart med live data fra serveren til Cordel.

4.5.2 Brukergrensesnitt – GUI

GUI'en er laget med tanke på brukervennlighet, hva arbeidsgiveren synes er viktig å ha med. Denne web applikasjonen skal i hovedsak bare brukes av utviklere som er ansatt i Cordel, så noe av designet vil være rettet mot det.

4.5.2.1 NavMenu

NavMenu'en itererer over alle komponentene som er i webapplikasjonen. I bunn og grunn ligger det en React-bootstrap(fra kap. 3.2.2) som er en veldig fleksibel

bootstrap i React sitt bibliotek.

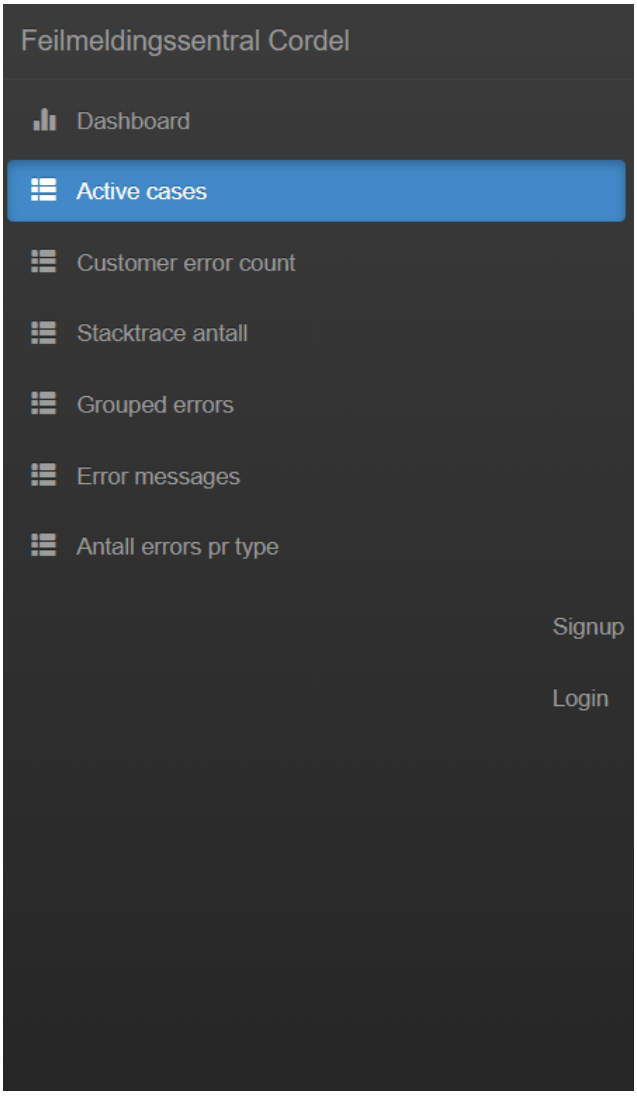
4.5.2.1.1 Design trekk som er blitt brukt

- Nærhet,
  som er et av trekkene i gestaltlovene nevnt i kap. 2.4
  Setter tabellene nært hverandre, mens autorisering komponentene for seg selv lengre ned på menyen.

- Likhet, som er et av trekkene i gestaltlovene navnt i kap. 2.4
  Tabellene har samme glyph fordi de inneholder mye av lik konstruksjon, mens Dashboard'et som inneholder en graf har fått en egen figur som tilsier at det skal være en grafisk figur i stedet for en tabell.

- Sammenheng
  Har også prøvd å skape en sammenheng mellom tabellene, der de tabellene som har samme ikon skal inneholde det samme. F.eks. så er det flere tabeller og de har alle samme ikon.

- Synlighet
  Designet til applikasjonen er veldig simpelt, men jeg har prøvd å lagt mest mulig vekt på synlighet, slik at brukeren skal finne frem til hva som trengs uten å måtte fundere på hvor ting er. NavMenu'en har et simpelt design som er lett for en hver bruker å se hvordan strukturen er satt sammen.



Figur 22 NavMenu til webapplikasjonen

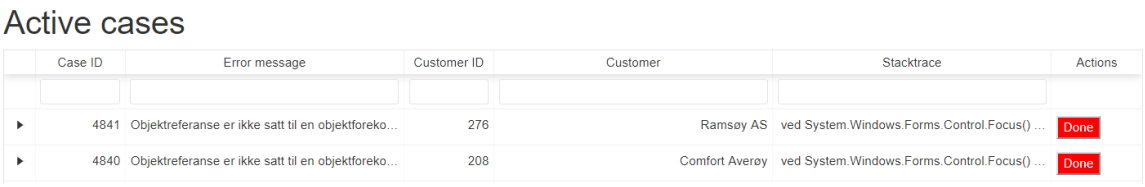
4.5.2.2 Tabell

Tabellen er den viktigste funksjonaliteten til webapplikasjonen og er derfor den som har blitt brukt mest tid på. Det er totalt 6 tabeller på webapplikasjonen som alle viser dataen på forskjellige måter.

Brukeren skal kunne:

- Se data live fra databasen
- Filtrere over dataen for å finne akkurat den feilmeldingen som er ønskelig
- Sortere etter ønskelig data
- Å finne utvidet informasjon om feilmeldingen.
- Å sette feilmeldingen til «Done» noe som skal fjerne den/dem fra tabellen. Dette skjer med at isActive klassen i error_message tabellen blir satt til false.

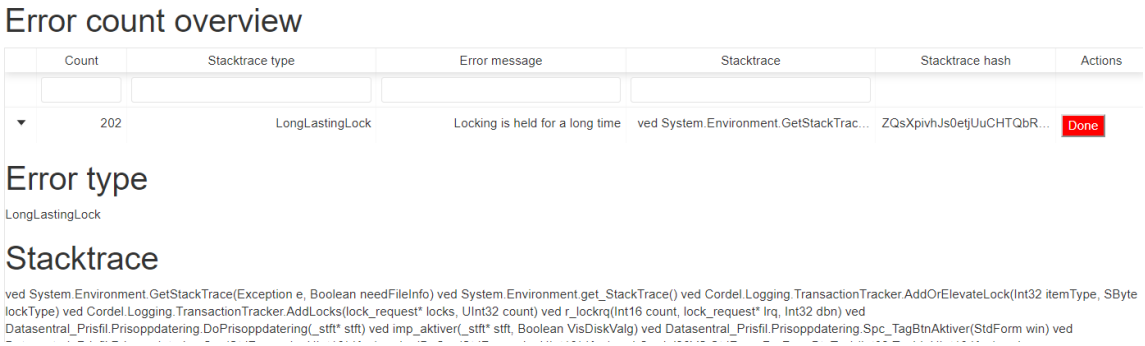
Databasen er satt opp med constraints slik at det bare er mulig å se data som har isActive som true/sant.



Figur 23 Tabell strukturen til webapplikasjonen

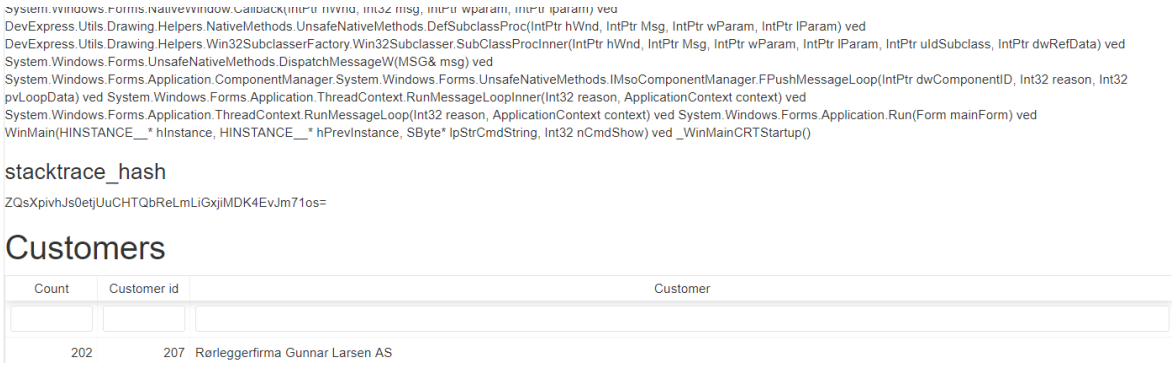
4.5.2.3 Master detail tabell funksjon

Error count overview tabellen viser alle feilmeldingene med samme type feilmelding. På eksempelet ser vi en LongLastingLock feilmelding som oppstår 202 ganger. Alle tabellene inneholder en såkalt subcomponent der det er mulig å se hele stacktracen og alt av relevant informasjon



Figur 24 Error count overview tabell

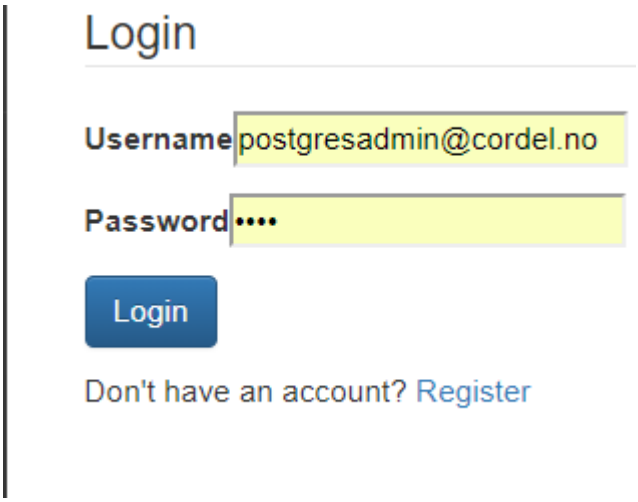
Det som er unikt for Error count overview tabellen er at den har en master detail med en esktern tabell der en kan se hvilke kunder som har fått feilmeldingen. På eksempelet ser vi at det er en kunde som har fått den samme feilen alle 202 gangene, men det kan også være 202 forskjellige kunder.



Figur 25 Master detail med ekstern tabell

4.5.2.4 Login

Innlogging var ingen prioritet, men ble laget for min egen skyld siden Cordel hadde en bruker tabell tilgjengelig. Grunnen til at jeg laget denne form'en var mer å lære meg å bruke React på en god måte.

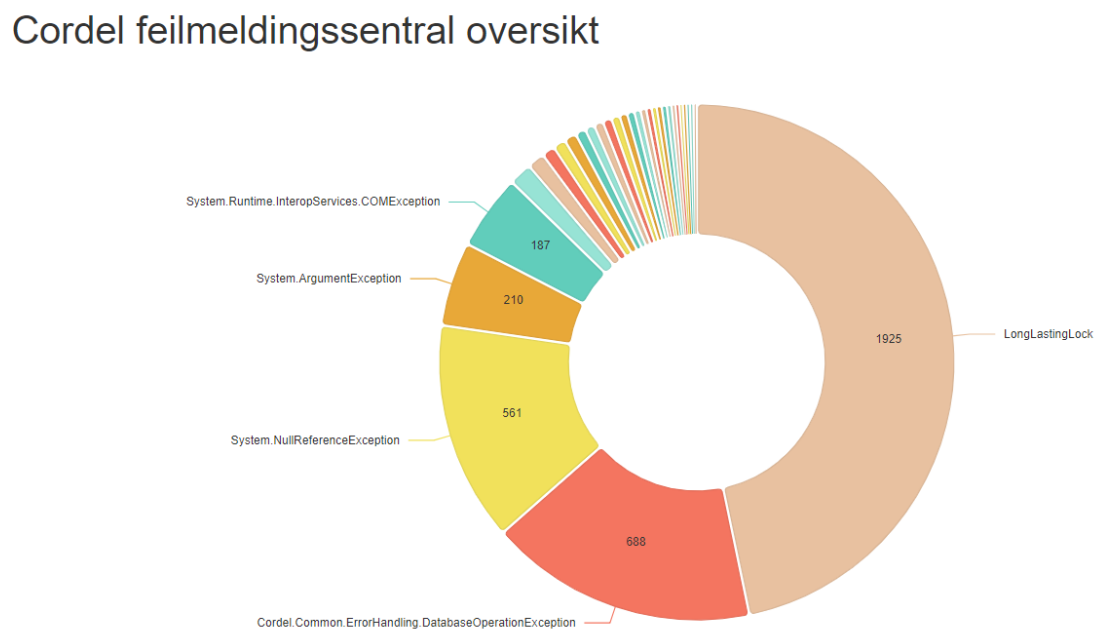


Figur 26 Innloggingen til webapplikasjonen

4.5.2.4.1 Design trekk som har blitt brukt

4.5.2.5 Dashboard med piechart

Applikasjonen har et dashboard som viser live data fra databasen til Cordel. De ønsket et piechart som itererer over alle de forskjellige stacktrace'ene som har kommet inn i systemet og er aktive. Laget her et pie chart ved hjelp av Nivo. Nivo er et react bibliotek som jeg nevnte tidliger(Kap. 3.2.2) og kan enkelt bli lagt til ved hjelp av Node.js sin pakkehåndteringsmetode.



Figur 27 Dashboard med Pie chart til webapplikasjonen

4.6 Dokumentasjon og utviklingsmetode

For å dokumentere prosjektet, så valgte jeg å bruke JIRA, Confluence og Github. Jeg er alene om prosjektet, men føler det ville skape en viss struktur i iterasjonene, samtidig som jeg ble enig med bedriften om å ha en ukes iterasjoner, der vi hadde møte hver torsdag fram til prosjektperioden sluttet.

- Jira ble brukt for å holde kontroll på de forskjellige oppgavene som skulle gjøres under sprint perioden.
- Confluence for å lage en oversikt over prosjektet sin helhet og for å lage møte rapporer med veileder
- Github ble brukt for å holde kontroll på kildekoden under prosjekt perioden.

5. Diskusjon

Her i denne seksjonen vil jeg diskutere og reflektere rundt de forskjellige valgene som har blitt tatt gjennom prosjektperiode. Blant annet valg av design, bibliotek, utviklingsmetoder. Vil også nevne hvilke problemer jeg opplevde, hvilke forbedringer jeg ville gjort om det var mer tid.

5.1 Kilder

Kilder er ikke alltid troverdig, så jeg brukte mye tid på starten for å finne informasjon med troverdige kilder. Fant noen bøker, troverdige nettsider, Youtube videoer samtidig som det var noen troverdige blogger med forfattere som har blitt bekreftet av flere at er troverdige.

I starten brukte jeg også lang tid på å velge hvilke bibliotek jeg ville bruke og byttet ofte mellom hvilke kilder og hvilke løsninger jeg ville bruke i forhold til hva jeg kunne finne av informasjon som passet prosjektmetoden jeg ville bruke. For eksempel så er det mange som bruker React med axios sitt bibliotek eller React med Redux. Dette fikk meg til å endre min applikasjon utallige ganger før jeg bestemte meg for å bruke React vanilla med fetch.

Jeg brukte også flere forskjellige bacheloroppgaver som har troverdige kilder og god informasjon om forskjellige metoder og verktøy som kan brukes til å utvikle en god løsning. Jeg fant til slutt en løsning som mener passet prosjektet mitt godt og er meget fornøyd med utviklingsmetoden som jeg endte opp med til slutt.

Når et bibliotek eller en metode er relativt ny, så er det viktig å bruke tid når en velger metoder og kilder før starter på prosjektet. Ta for raske beslutninger ang. kilder kan ha en stor påvirkning på sluttresultatet.

5.2 Valg av utviklingsmetoder

5.2.1 Rammeverk/bibliotek

Valget av rammeverk/bibliotek var det i prosjektet som jeg brukte lengst tid på å velge ut. Grunnen til dette er en avgjørende fase når det gjelder kodestil og hvordan applikasjonen din vil bli bygd opp på klient siden.

5.2.1.1 Sammenligning av React, Vue og Angular

Alle tre er utmerkede verktøy til å lage en singel-page webapplikasjon på hver sin måte. Der React og Vue er mer fleksible, mens Angular har mer en strengere syntaks og er noe mer komplisert å lære seg, men når du først har forstått det, så er det en høy læringskurve.

- Vue.js
  Vue sin største fordel er at den er veldig liten. Dette gjør at det er et veldig raskt og fleksibelt rammeverk. Et av de mest negative delene med Vue.js er at markedet er så lite. Det er ikke mange som bruker rammeverket. Dette gjør at det er ikke veldig mange kilder eller tidligere applikasjoner å se til. Noe av grunnen til dette er at det er såpass nytt som det er, så det er mye mulig dette vil komme med tiden.

- Angular
  Angular sine største fordeler er at dokumentasjonen deres er veldig godt dokumentert, samtidig som syntaksen gir deg en veldig god kodestruktur med MVVM som gjør det mulig for utviklere å jobbe med samme applikasjon med samme samme sett av data. Det som er negativt med Angular er at syntaksen kan være ganske avansert. Angular bruker Typescript som har noe strengere syntaks enn vanlig ES6 Javascript.

- ReactJS
  ReactJS er det som er mest brukt i disse dager og noe av grunnen for det er akkurat den grunnen til at jeg valgte akkurat denne utviklingsmetoden for webapplikasjonen. De største fordelene for ReactJS er at den er lett å lære. Om en forstår Javascript fra før, så tar det ikke lang tid før en kan bruke ReactJS og Javascript sammen. Det er også mulig å bruke Typescript og ReactJS sammen, men dette er noe

jeg ikke ønsket å gjøre. Grunnen til det er fordi jeg ville oppnå så mye fleksibilitet som mulig. Jeg hadde ikke mye erfaring med Javascript når jeg startet med dette prosjektet, så da føltes det mest naturlig å velge mest mulig fleksibilitet.

5.2.2 Programmeringsspråk

Med React sitt bibliotek, så er det i hovedsak vanlig å bruke Javascript, men det er også mulig React med Typescript. Grunnen til at jeg valgte dette var fordi jeg hadde mest erfaring med Javascript fra før av og det er mye mer brukt enn Typescript, så jeg følte Javascript var den mest fornuftige språket å bruke.

5.2.3 PostgreSQL og PostgREST

Siden jeg var alene på prosjektet, så bestemte Cordel seg for å bruke PostgREST som et perfekt mellomledd mellom database og React. Fikk også tilgang til databasen ved hjelp av pgAdmin 4 og VPN, som nevnt i kapittel 3 og hadde god oversikt over de fysiske tabellene samt. Kunne lage Views tilpasset til prosjektet.

5.2.4 Confluence, JIRA og Github

Grunnen til at jeg valgte å bruke nettopp disse verktøyene er mye på grunn av tidligere kjennskap til verktøyene gjennom skolen. Samtidig så er min veileder, Arne Styve administrator på Confluence og JIRA, så dette gjorde at jeg fikk en god og ordentlig gjennomgang av mulighetene før jeg startet å bruke det.

Github har jeg brukt mye før gjennom forskjellige fag på skolen og jeg ser på det som en veldig ordentlig og ryddig måte å holde styr på koden. Dette er noe som spesielt trengs om en er flere personer som jobber sammen, men det kan også være greit for en som jobber alene mtp. Å gå tilbake på evt. Feil som måtte oppstå.

5.3 Refleksjon

Om jeg skal se tilbake på når jeg startet med prosjektet og reflektere over hva jeg kunne gjort annerledes, så vil jeg si at valg av utviklingsmetode i React var det jeg brukte mest tid på og skulle nok vært mer konsekvent på valgene mine av bibliotek jeg valgte å bruke. Jeg prøvde først med axios, men følte ikke den passet noe særlig til applikasjonen. Prøvde også noe Redux, men valget falt til slutt på bruk av fetch og vanilla javascript, noe som jeg mener var den riktige beslutningen, men brukte da en god del tid på å finne ut av dette og tid er en mangelvare når en skal skrive bachelor.

5.3.1 Estimering

Er det noe jeg virkelig skulle vært bedre på i dette prosjektet, så er det estimering av hvor mye tid jeg skal bruke på hver enkelt element. Estimering i programmeringsbransjen er alltid vanskelig, og jeg er derfor jeg brukte flere elementer av agile metoder.

Da programmeringsdelen begynte virkelig å ta fart, så var det plutselig på tide å skrive bachelor, så en ordentlig estimering av når en må slutte å programmere og si seg fornøyd med applikasjonen for så å begynne på rapporten hadde kanskje gjort det mindre hektisk mot slutten.

5.3.2 Typescript

Typescript bringer flere fordeler når en programmerer en webapplikasjon, men det kan også være ganske vanskelig, når en ikke har veldig mye Javascript erfaring fra før av. Jeg startet med å programmere i Typescript, men fant kjapt ut at dokumentasjonen jeg fant var for det meste for Javascript, så jeg valgte å holde meg til det.

5.4 Problemer

Etter hvert som prosjektet begynte å komme litt nærmere sluttdato, så ser en alltids at det er noen ting som en ikke vil ha tid til å implementere, noe som en ikke kommer til å bli helt fornøyd med eller rett og slett noe du skulle løst på en bedre måte enn det som ble gjort.

5.4.1 Valg av bibliotek

I starten var det vanskelig å vite hvilke bibliotek som var de beste å bruke for videreutvikling av applikasjonen. Det måtte en god del undersøkelser til for å finne de React bibliotekene som passet til applikasjonen sitt formål. En kunne f.eks. bruk Redux sitt bibliotek, men da ikke er nødvendig når dette ikke er snakk om en veldig stor applikasjon.

5.4.2 Kodestil

5.4.3 React

React er ikke like bra dokumentert som f.eks. Angular, så det er ikke like lett å lese seg opp på dokumentasjonen om hvordan ting fungerer i React. Etter hvert fant jeg flere youtube videoer som passet bedre enn dokumentasjonen.

5.5 Forbedringer

5.5.1 Design

5.5.2 Autorisering

Autorisering var som tidligere nevnt ikke en av prioriteringene iom. At applikasjonen bare skulle bli brukes internet i bedriften. En login og registrering form ble laget, men implementasjonen ble ikke helt som jeg ønsket, så autorisering er ingen nødvendighet ved å bruke applikasjonen per dags dato.
Det som da mangler for at autorisering skal være klart:

- JWT sikkerhet
- Litt endringer i POST metoden

5.5.3 Splitting av UI

Strukturen på koden kunne vært bedre, da jeg startet med prosjektet når jeg ikke var veldig inn i React. Et av React sine fordeler er å bruke opp igjen kode og splitte kode inn i mindre komponenter. Noen komponenter ble noe store noe som ble noe

problematisk etter hvert. En forbedring kunne ha vært der alt av http kall kunne vært i en og samme component.

Grunnen til at det ble slik, var fordi jeg startet med å programmere før jeg hadde en full forståelse av hvordan «best practice» i React var. Og hvordan det er vanlig å strukturere koden.

6. Konklusjon

Hensikten med denne bacheloroppgaven var å lage en webapplikasjon som skulle lage en web applikasjon med et feil-rapporteringssystem som kan:

- Hente ut tilstrekkelig med informasjon fra serveren
- Vise dataen til brukeren
- Stort fokus på effektivisering hos brukeren, søkefunksjoner og sortering data
- Gjøre det mulig å håndtere data med forskjellige CRUD operasjoner gjennom http og REST

Applikasjonen jeg endte opp med har alle disse punktene nevnt ovenfor og jeg kan si meg fornøyd med resultatet jeg har oppnådd i denne perioden. Selv hadde jeg ingen erfaring om React fra før og hadde heller ingen store Javascript kunnskaper, så dette har vært en veldig lærerik periode for meg. Dette er da ingen fasit på hvordan en skal lage en webapplikasjon og det er sikkert en god del som kan ha blitt gjort for å få et bedre og raskere resultat, men jeg er alt i alt fornøyd med produktet som jeg endte opp med og samarbeidet med Cordel har vært lærerikt, morsomt og interessant på alle måter.

NTNU i Ålesund

Bacheloroppgave

Oppsummering av Cordel:

«Studenten har utført et prosjekt som gikk ut på å lage en web frontend til visning av data på en PostgreSQL-database som Cordel Norge as bruker til feil-rapportering.

Han har brukt PostgREST og React som de to byggesteinene i prosjektet. I tillegg ligger PostgreSQL-databasen i bunnen og som Cordel -produktene leverer sine rapporter til. Han har i dette arbeidet fått innsyn i SQL, React og JavaScript samt i Visual Studio Code.

Jira har vært brukt for å organisere arbeidet og jeg har kunnet følge med på status på de ukentlige statusmøtene vi har hatt.

Som versjonshåndteringssystem har han brukt GIT og fått et innblikk i hvordan dette fungerer.

Hovedmålet for prosjektet er løst tilfredsstillende sett fra vår side.

Samarbeidet med studenten har fungert godt.»

7. Referanser

1. Postman [Internett]. Postman. [Hentet 2018 Nov. 12]. Tilgjengelig her: https://www.getpostman.com/
2. Swagger [Internett]. Swagger. [Hentet 2018 Nov. 12] Tilgjengelig her: https://swagger.io/
3. PgAdmin [Internett]. pgAdmin. [Hentet 2018 Nov. 12] Tilgjengelig her: https://www.pgadmin.org/
4. React developer tool [Internett]. React developer tool [Hentet 2018 Nov. 12] Tilgjengelig her: https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en
5. VPN SonicWall client [Internett] DELL SonicWall GLOBAL VPN client[Hentet 2018 Nov. 12] Tilgjengelig her: https://www.sonicwall.com/en-us/home
6. Jira [Internett] Jira[Hentet 2018 Nov.12] Tilgjengelig her https://www.atlassian.com/software/jira
7. Confluence[Internett] Confluence[Hentet 2018 Nov. 12] Tilgjengelig her: https://www.atlassian.com/software/confluence
8. Github[Internett] Github [Hentet 2018 Nov. 12] Tilgjengelig her: https://github.com/
9. Javascript[Internett] Javascript[Hentet 2018 Nov. 13] Tilgjengelig her: https://javascript.info/intro
10. Git[Internett]Git[Hentet 13 Nov.] Tilgjengelig her: https://git-scm.com/
11. Microsoft Visual Studio[Internett] Visual Studio [Hentet 13 Nov.] Tilgjengelig her: https://visualstudio.microsoft.com/
12. ReactJS[Internett] React [Hentet 13 Nov.] Tilgjengelig her: https://reactjs.org/
13. Node.js[Internett] Node[Hentet 13 Nov.] Tilgjengelig her: https://nodejs.org/en/
14. JsonView[Internett] JsonView[Hentet 13 Nov.] Tilgjengelig her: https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc?hl=en
15. Hypertext transfer protocol[Internett] http [Hentet 18.Nov] Tilgjengelig her: https://www.techopedia.com/definition/2336/hypertext-transfer-protocol-http

16. RESTful web services[Internett] RESTful web services[Hentet 19. Nov] Tilgjengelig her: https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html
17. JSON[Internett] JavaScript Object Notation[Hentet 19.Nov] Tilgjengelig her: https://www.json.org/
18. Wireframes[Internett] Wireframes[Hentet 19.Nov.] Tilgjengelig her: https://www.usability.gov/how-to-and-tools/methods/wireframing.html
19. NordBø T. Introduksjon til Interaksjonsdesign. :4
20. Singel-page applikasjon [Internett] Singel-page applikasjon[Hentet 20.Nov.] Tilgjengelig her: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx
21. postgREST[Internett] postgREST[Hentet 20.Nov.]Tilgjengelig her: https://www.postgrest.org
22. HTML[Internett] Hypertext Markup Language[Hentet 20.Nov.] Tilgjengelig her: https://www.techopedia.com/definition/1892/hypertext-markup-language-html
23. HTML & CSS[Internett] Cascade Style Sheet [Hentet 20.Nov.] Tilgjengelig her: https://www.w3.org/standards/webdesign/htmlcss
24. PostgreSQL[Internett] PostgreSQL[Henter 20.Nov.] Tilgjengelig her: http://www.postgresqltutorial.com/what-is-postgresql/
25. PostgreSQL Views[Internett] Views [Hentet 25.Nov.] Tilgjengelig her: http://www.postgresqltutorial.com/managing-postgresql-views/
26. VueJS[Internett] Vue [Henter 26.Nov.] Tilgjengelig her: https://vuejs.org/v2/guide/
27. Angular[Internett] Angular [Hentet 26.Nov.] Tilgjengelig her: https://www.sitepoint.com/angular-introduction/
28. Typescript[Internet] Typescript [Hentet 26.Nov.] Tilgjengelig her: https://www.educba.com/typescript-vs-javascript/
29. Webapplikasjon «layers»[Internett] Oppbyggingen til en webapplikasjon[Henter 27.Nov.] Tilgjengelig her: https://hackernoon.com/architecting-single-page-applications-b842ea633c2e
30. Web applikasjonens arkitektur historie[Internett] Webapplikasjonens historie[Hentet 27.Nov.] Tilgjengelig her: https://blog.octo.com/en/new-web-application-architectures-and-impacts-for-enterprises-1/
31. Codd EF. A relational model of data for large shared data banks. Communications of the ACM. 1970;11

32. Sommerville I. Software engineering. Tenth edition, global edition. Boston, Mass. Amsterdam Cape Town: Pearson Education Limited; 2016. 810 p. (Always learning).
33. Fielding RT. Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine; 2000.
34. dbDiagram [Internett] dbDiagram [Henter 27.Nov.] Tilgjengelig her: https://dbdiagram.io/
35. Sandnes EF. Universell utforming av IKT-systemer. 2.utgave; 2018. Kap. 3.3(Gestaltlovene)
36. Cordel [Internett] Cordel [Hentet 28.Nov.] Tilgjengelig her: www.cordel.no
37. React-router-bootstrap[Internett] React-router-bootstrap[Hentet 29.Nov.] Tilgjengelig her: https://www.npmjs.com/package/react-router-bootstrap
38. Nivo[Internett] Nivo/pie [Hentet 29.Nov.] Tilgjengelig her: https://www.npmjs.com/package/nivo
39. React-table[Internett] React-table [Hentet 29.Nov.] Tilgjengelig her: https://react-table.js.org/
40. React-bootstrap[Internett] React-bootstrap [Hentet 29.Nov.] Tilgjengelig her: https://www.npmjs.com/package/react-bootstrap
41. UML[Internett] Unified Modeling Language [Hentet 29.Nov.] Tilgjengelig her: https://www.visual-paradigm.com/guide/uml-unifiedmodeling-language/what-is-uml/

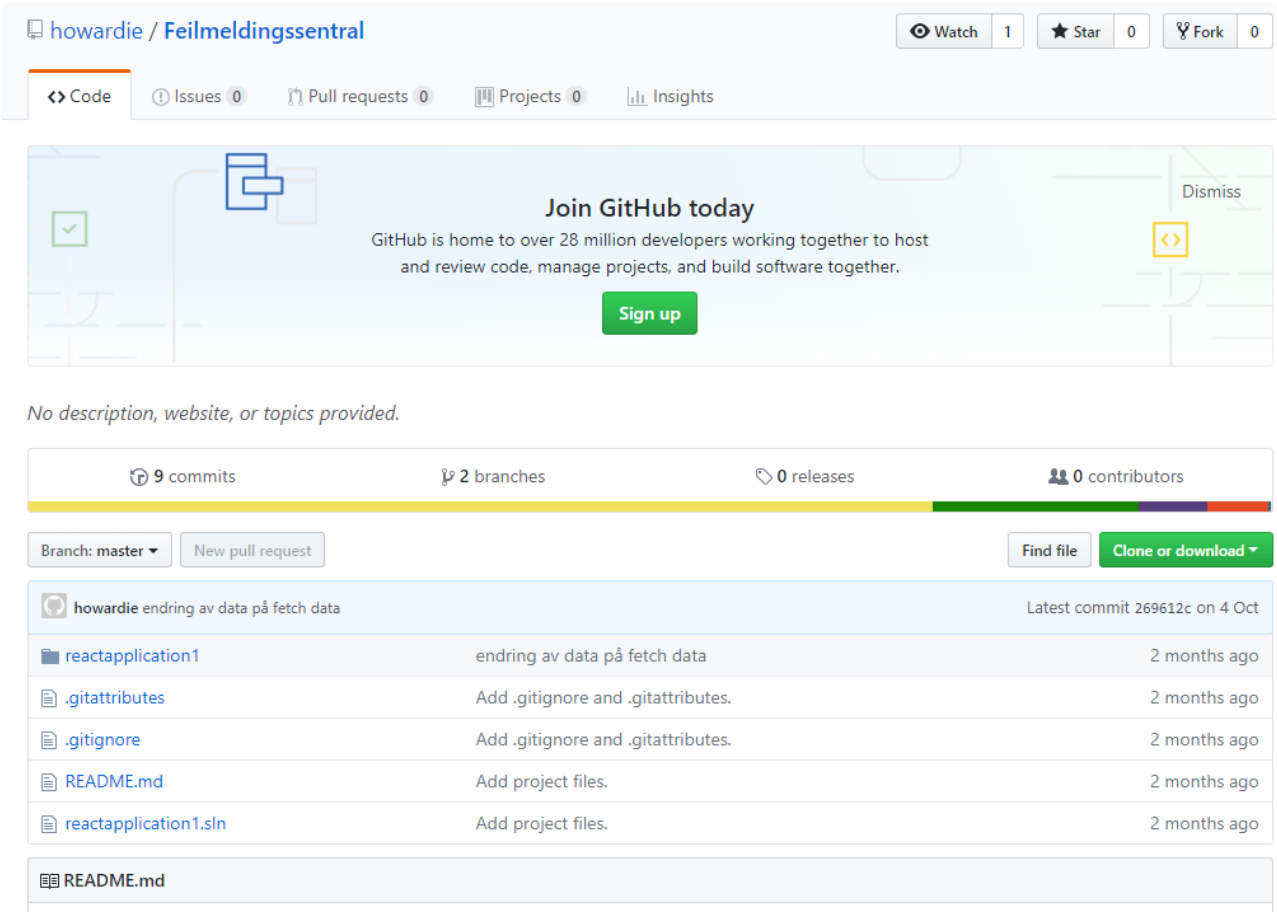
NTNU i Ålesund

Bacheloroppgave

8. Vedlegg 1: Kildekode

Kildekoden min finnes på Github under min egen bruker:
https://github.com/howardie/Feilmeldingssentral

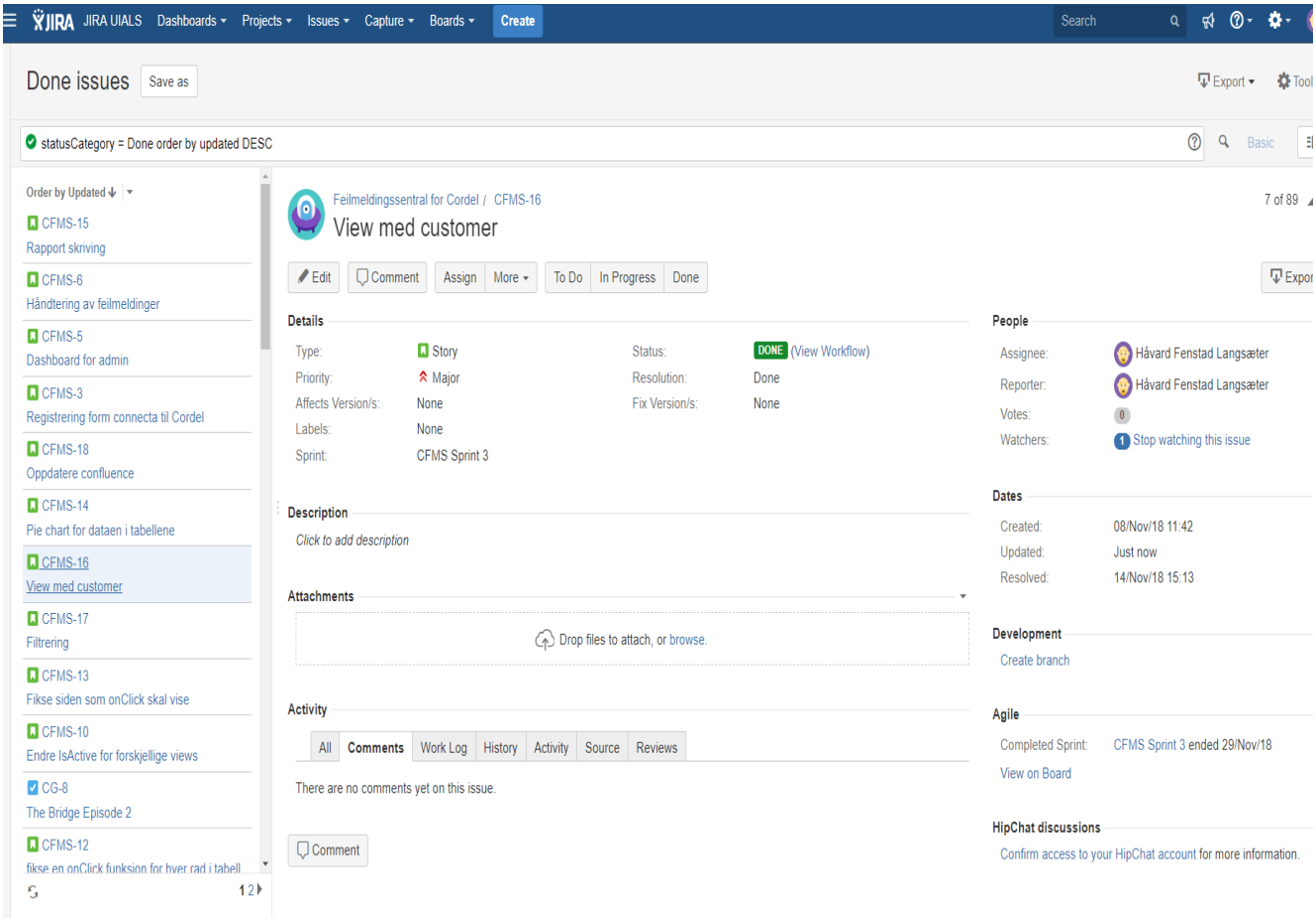
Kildekoden min har også blitt levert i en egen zip fil.



NTNU i Ålesund

Bacheloroppgave

9. Vedlegg 2: JIRA



NTNU i Ålesund

Bacheloroppgave

NTNU i Ålesund

Bacheloroppgave

10. Vedlegg 3: Forprosjektrapport

INNLEDNING

Grunnen til at jeg valgte denne oppgaven var fordi oppgaven virket overkommelig og spennende. Samtidig har jeg hørt veldig mye bra om Cordel som bedrift. Oppdragsgiveren er som tidligere nevnt Cordel. Cordel har en lokal avdeling her i Ålesund, men har kunder over hele landet. Bakgrunnen for prosjektet er at Cordel ønsker en mer oversiktlig og effektiv måte for deres ansatte å håndtere feilmeldingene kundene sender inn. I søk om dette, så vil forhåpentligvis web applikasjonen være til stor hjelp med dette. Hele formålet med prosjektet er at web applikasjonen skal være en viktig brikke i hverdagen til mange av de ansatte på Cordel.

BEGREPER

Javascript:

- Er et programmeringsspråk som har en høy level. Det vil si at det ligger som regel på toppen av programmeringen og blir ofte sett av brukeren.

Typescript:

- Er et programmeringsspråk som er utviklet som et mer strikt supersett av Javascript.

Github:

- Er en web basert versjon kontroll system, som gjør det mulig å bruke Git inne på mine IDE'er. Git brukes for å ha kontroll på koden. Om det brukes riktig, så skal det være lett å gå tilbake til siste endring som ble lastet opp.

Agile metoder:

- Eller smidige metoder, som kanskje ville vært enda mer fornorsket er hvordan en prosjektprosess skal foregå. Smidige metoder er en reaksjon på den tradisjonelle prosjektoppbyggingen.

NTNU i Ålesund

Bacheloroppgave

AVTALER

Avtale med oppdragsgiver

Har til nå en muntlig avtale med Cordel om å utvikle en web applikasjon som passer deres behov. Skal etterhvert skrive en avtale mellom student,bedrift og NTNU om hvordan prosessen skal foregå.

Arbeidssted og ressurser

Arbeidsplass:

Vil hovedsaklig bruke data-labben(L167) som arbeidsplass ellers vil jeg jobbe hjemmefra, da jeg er alene.

Ressurser:

Cordel har gitt meg tilgang til databasen og kan teste data uansett hvor jeg måtte være.

Datasikkerhet:

Har/skal skrive under en klausulsavtale angående evt. bedriftsinformasjon til Cordel.

Tilgang til personer:
Har møte med prosjektansvarslig hos Cordel hver uke. Han responderer også på mail til en hver tid.

Avtalt rapportering:
Har til nå vist en kravspesifikasjon til bedriften og leverer en statusrapport til Cordel hver uke.

PROSJEKTBESKRIVELSE

Problemstilling- målsetting- hensikt

Resultatmål:

- Web applikasjonen skal kunne opprette brukere i databasen.
- Skal være mulig å logge inn med passord og brukernavn.
- Web applikasjonen skal ha en oversiktlig og god gui.
- Web applikasjonen skal iterere over forskjellige produkter som ligger inne.
- Skal være mulig å se hvilke feilmeldinger som er håndtert på de forskjellige produktene.

Effektmål:

- Skal ha en bedre oversikt over feilmeldingene som kommer inn en det er i dag.
- Skal effektivisere hverdagen til en ansatt som håndterer de forskjellige feilmeldingene.

Krav til løsning eller prosjektresultat – spesifikasjon

Det er ikke satt noen formelle krav ang. Hvor mye som må fungere, men ser helst at det skal være en fungerende web applikasjon som kan ta i mot og håndtere informasjon fra deres database på en ordentlig måte. At det er en fungerende feilmelding sentral på en eller annen måte har vi blitt enig om som et krav.

Fullført prosjekt vil si at det er en fungerende feilmelding sentral. Det vil alltids bli ting som kan bli lagt til, men alt dette blir gjort etter hvor langt tiden strekker til.

Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)

Jeg har kommet fram til at agile metoder er den beste løsningen for dette prosjektet, der alle involverte parter har et tett samarbeid om hva som blir gjort over korte itterasjoner. Denne metoden er veldig bra for alle parter. Med mye informasjon, så kommer ingen overaskelser mot slutten og reflekteringsfasen blir mye lettere.

Informasjonsinnsamling – utført og planlagt

Har ikke mye erfaring med bruk av Javascript/Typescript og React som rammeverk, så dette må undersøkes og læres på en ordentlig og god møte før en kan sette i gang med den viktige programmeringsdelen. Jeg har planlagt å bruke en del tid på guide og slikt fra internett til å håndtere de forskjellige metodene som trengs.

Alt av informasjon som blir hentet er gjennom internett og bedriften Cordel. Cordel har flere personer som er gode på javascript og har satt seg noe inn i react, så mere spørsmål til dem, samt. Flere søk på internett gjennom resten av prosjektet.

Vurdering – analyse av risiko

Vurdering av muligheten for å realisere prosjektet innenfor den rammen som er gitt.
Vurdering av behov for og forslag til eventuell ytterligere presisering og/eller avgrensning av prosjektet som ennå ikke er avklart.

Risikoen for at prosjektet ikke blir gjennomført er minimal. Web applikasjonen sine hovedtrekk vil bli gjennomført i de første prosessene og deretter vil det etterhvert bare stå på en forbedringsprosess, som bedriften selv kan gjøre perfekt om ikke tiden min skal være tilstrekkelig nok. De vil uansett jobbe videre med web applikasjonen etter at jeg er ferdig.

Hva som skal til for å lykkes? For at prosjektet skal bli velykket, så må jeg mestre rammeverket som jeg har valgt å bruke, samtidig som javascript/typescript må kunne håndteres på en grei måte for å oppnå et godt gjennomført prosjekt.

Trusselen for prosjektet sin del er at javascript/typescript er noe en har lært for det meste på egenhånd og veldig lite undervisning i, samtidig som det alltids kan oppstå en eller annen feil.

Rammeverket som er valgt er veldig nymoderne, noe som kan gjøre det litt vanskelig å finne mange eksemplarer av bruk og hjelp.

Hovedaktiviteter i videre arbeid

Estimering av hvor lang tid hver aktivitet vil ta er vanskelig mtp. at det er liten erfaring med de forskjellige verktøyene fra før og en feil i koden kan utsette ting betraktelig, men målet er å bli ferdig med alle fire punktene minimum for å kunne si prosjektet ferdig.

Nr	Hovedaktivitet
1.	Lage en fungerende web-gui
2.	Lage en kobling mellom databasen og web-guien
3.	Få hentet inn riktig informasjon fra databasen
4.	Lage flere funksjoner som effektiviserer bruken på mest mulig måte

Framdriftsplan – styring av prosjektet

Hovedplan

- Hovedtrekk i gjennomføringen
- Beskrivelse av planlagte hovedaktiviteter i forhold til listen under pkt 5.6, om nødvendig med kort beskrivelse av viktige underaktiviteter (hvilke, innhold, oppgavefordeling, ansvar, forventet startidspunkt, sluttidspunkt osv.)
  Milepæler (hva skal være oppnådd - når) – resultater / leveranser
  Beslutningsprosess - viktige beslutningspunkter (når, om hva, av hvem)

Prosjektet har fire faser som må gjennomføres stegvis før den neste kan begynnes på

1. Lage en basic web-gui med react
   - Lage innlogging og registrering
   - Gui for feilmelding sentralen
2. Koble databasen til web-guien
   - Lagt til brukere i databasen ved hjelp av web-gui
   - Logget inn med bruker fra databasen
   - Legge til produkt ved hjelp web-gui
3. Administrasjon på web-gui

- Dashboard over alle de forskjellige brukerne og muligens andre endringer

Styringshjelpemidler

Planen er å bruke noe lignende Jira, slik at en har klare mål for den ukentlige itterasjonen og hvilke oppgaver som må gjøres for at en skal ligge godt an i forhold til fristen. Selv om jeg er alene på prosjektet, så er det fremdeles greit å ha noe som kan hjelpe meg å ha en fin oversikt over hva som trengs å gjøres i hvilken rekkefølge.

Utviklingshjelpemidler

- Har flere forskjellige programmer og verktøy som jeg skal bruke for å utvikle prosjekte slik som rammeverket, React, Visual Studio til å programmere Javascript/Typescript i. Må også bestemme hvordan koblingen til databasen skal bli gjort.
- Bruker også versjon kontroll styringen Github for å ha kontroll på koden, slik at det er lett å gå tilbake til hvor jeg var om, eller ha en egen branch for test, slik at jeg ikke mister viktig kode på noen som helst måte.

Beslutninger – beslutningsprosess

Avgjørelser om hvilke utviklingsmetoder som skal bli brukt har jeg gjennomført sammen med Cordel. De sier hvilken formål de ønsker å oppnå og en pekepinn og jeg gjør undersøkelser om hva som vil passe meg best og dette er en veldig bra løsning for begge parter i dette tilfellet. Derfor blir dette prosjektet utviklet i rammverket react med bruk av Javascript/Typescript.

DOKUMENTASJON

Rapporter og tekniske dokumenter

- Koden jeg skriver vil bli dokumentert, slik at det blir lettere for bedriften å utføre vedlikehold eller endringer i fremtiden. Ellers vil bacheloroppgaven også dokumentere oppbyggingen av koden og hvorfor det har blitt gjort slik som det har blitt.

PLANLAGTE MØTER OG RAPPORTER

Møter

Møter med styringsgruppen

- Har mitt første møte med veileder, 02.10.18, 15:00.

Prosjektmøter

- Har møte med bedriften hver torsdag for å få rapportering om status og evt. Hjelper meg om nødvendig.

PLANLAGT AVVIKSBEHANDLING

Om prosjektet ikke går som planlagt, så vil endringer eller begrensninger bli tatt opp bedriften for å diskutere hva som er beste løsning. Dette er enkelt å håndtere da vi har mye kontakt via E-mail samtidig som vi har et møte hver uke.

Det er mitt ansvar å komme med forslag til nye endringer som evt. Må gjøres, men hovedsaklig vil jeg fokusere på å få prosjektet så likt somdet bedriften har som ønske å få på plass.

UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING

- Forutsetninger for at prosjektet skal bli gjennomført som det skal, er at jeg har en laptop/pc som er tilgjengelig og har alt av verktøy som trengs for å programmere web applikasjonen. Utenom det, så har jeg ikke særlig store behov for noen som helst andre verktøy for å gjennomføre, men dette kan endre seg utover i prosjektet.

NTNU i Ålesund

Bacheloroppgave

11. Vedlegg 4: Møtereferat med veileder

2018-10-18 Meeting notes

Date

18 Oct 2018

Attendees

- Arne Styve
- Håvard Fenstad Langsæter

Goals

- Statusmøte, få en gjennomgang av hvordan målene på sprinten blir gjennomført.
- Hvordan en confluence fremside skal være.

Discussion items

Time	Item	Who	Notes
10	Jira	Håvard Fenstad Langsæter	• Bruk av agile metoder i prosjektet
20	Status	Håvard Fenstad Langsæter	• Siste sprint forløp • Hvordan samarbeidet med Cordeler • Fremdrift • Rapport

Action items

NTNU i Ålesund

Bacheloroppgave

- Håvard Fenstad Langsæter Oppdater hovedsiden på wiki med riktig info om kontaktperson etc.
- Håvard Fenstad Langsæter: Oppratt prosjektplan på forside wiki (roadmap)

2018-11-01 Meeting notes

Date

05 Nov 2018

Attendees

- Håvard Fenstad Langsæter
- Arne Styve

Goals

- Bachelor rapporten og forskjellige spørsmål rundt den
- Få en oversikt over selve innleveringsprosessen

Discussion items

Time	Item	Who	Notes
10min	Bachelor rapport	Håvard Fenstad Langsæter	• Generell informasjon
5min	Innlevering	Håvard Fenstad Langsæter	▪ Hvordan prosessen er, skal det være likt som vårsemesteret?

NTNU i Ålesund

Bacheloroppgave

2018-11-22 Meeting notes

Date

22 Nov 2018

Attendees

- Håvard Fenstad Langsæter
- Arne Styve

Goals

- Gjennomgang av divese informasjon ang. innlevering.
- Videreformidle diverse spørsmål om sluttdato, fremføringstidspunkt osv.
- Få en gjennomgang av hva som er viktig å ha med i en rapport og samtidig kanskje få tips om hva som kan gjøres bedre.

Discussion items

Time	Item	Who	Notes
10min	Innlevering	Håvard Fenstad Langsæter	• Informasjon
15min	Rapport	Håvard Fenstad Langsæter	Generelle spørsmål om rapportskriving ▪ Tips og triks ▪ Eksempel på en god oppgave

NTNU i Ålesund

Bacheloroppgave

12. Vedlegg 5: Confluence dokumentasjon

Feilmeldingssentral for Cordel
About this project

Cordel ønsker en applikasjon som skal iterere over alle feilmeldingene de får fra kundene sine. Det skal være mulig å få en oversikt over alle feilmeldingene på forskjellige måter. Det skal også være søkemuligheter,Charts og annen funksjonalitet som kan hjelpe på for å gi en god oversikt.

Et tett samarbeid mellom Cordel og utvikler vil forhåpentligvis gi et godt resultat til slutt som begge parter kan stille seg bak og være fornøyde med.

Student

Studenten som skal utføre denne oppgaven heter **Håvard Fenstad Langsæter** og studerer dataingeniør ved NTNU Ålesund.

Kontaktpersoner hos Cordel:

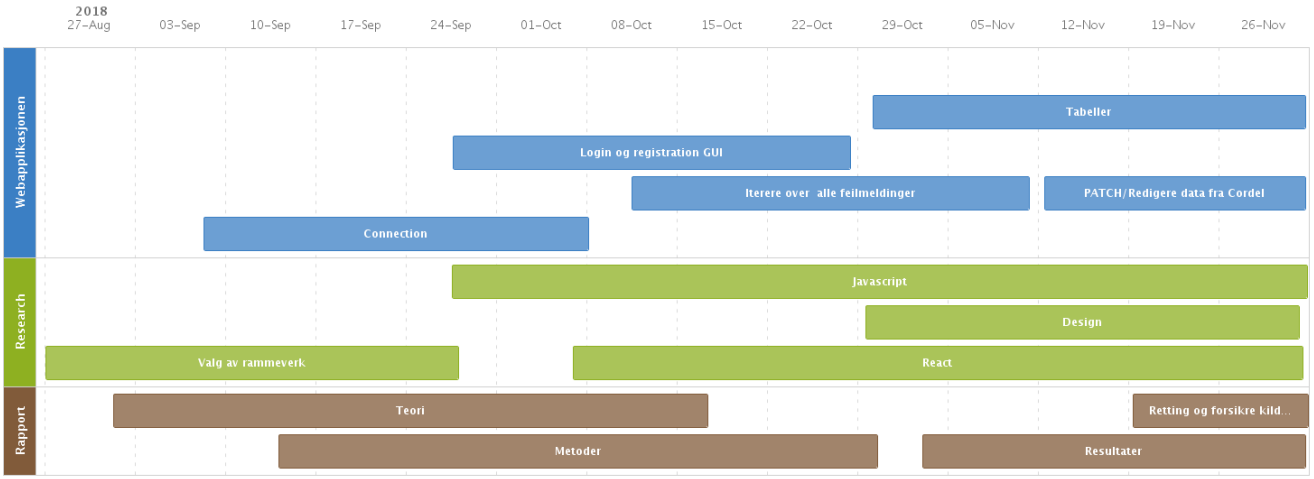
Ole Johan Larsen (Ole@cordel.no)

Harald Bjørshol (Harald@cordel.no)

NTNU i Ålesund

Bacheloroppgave

Roadmap



Vil også legge ved Confluence siden til prosjektet, som en evt. Kan gå inn og få et bedre bilde om muligheten her der:

http://confluence.uials.no:8090/pages/viewpage.action?spaceKey=CFMS&title=Feilmeldingssentral+for+Cordel

NTNU i Ålesund

Bacheloroppgave

NTNU i Ålesund

Bacheloroppgave