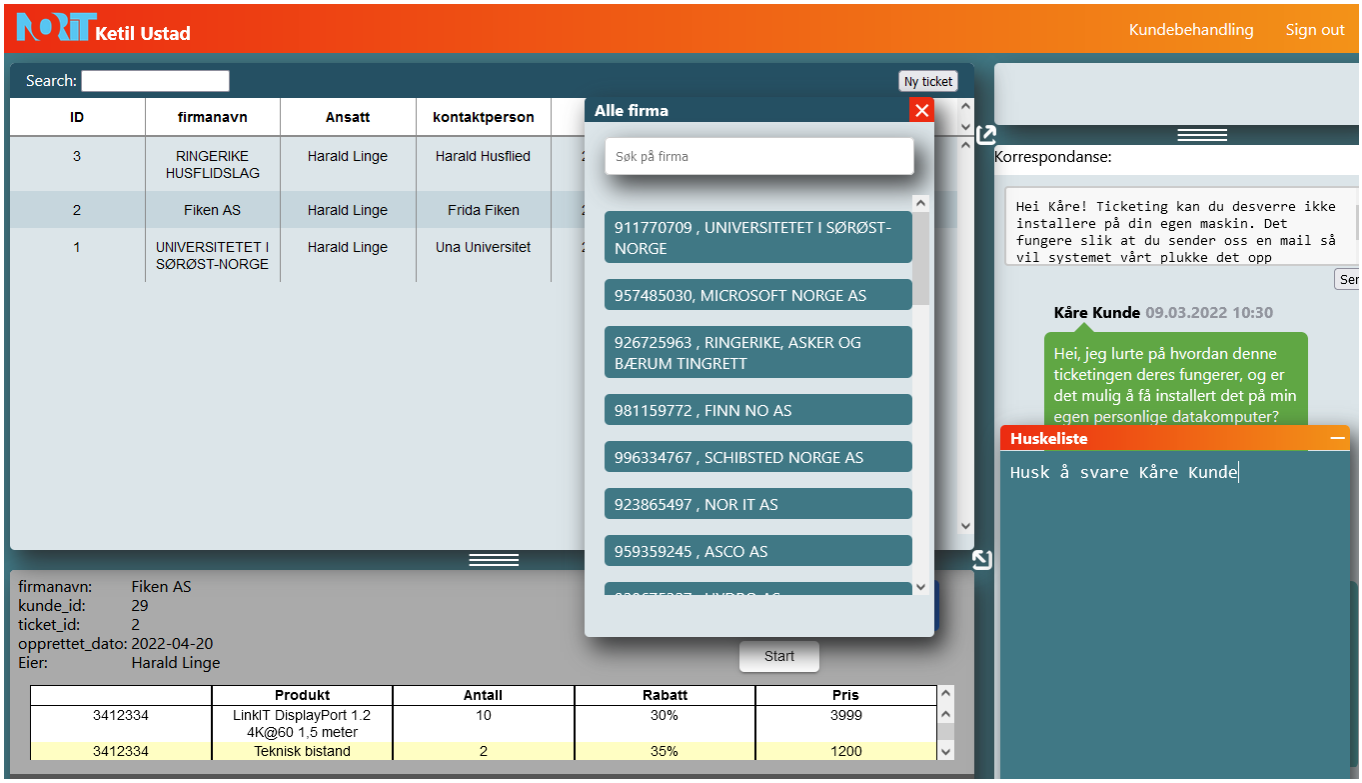
Universitetet i Sørøst-Norge

BOP3000

Bacheloroppgave i IT og Informasjonssystemer

Professional Service Automation verktøy for Nor IT



Forord

Rapporten er en del av Bachelor oppgaven vår for våren 2022 og den avsluttende oppgaven for studie IT og informasjonssystemer ved Universitetet i Sørøst-Norge (USN), Campus Ringerike. Under prosjektet har vi både benyttet oss av erfaring og kunnskap vi har tilegnet oss under studiet, men vi har også fått ny kunnskap om både ulike teknologier og prosjektstyring. Dette har vært både spennende, tidkrevende og vanskelig, men ikke minst svært lærerikt. Teamet har jobbet godt sammen igjennom hele perioden og vi leverer både et produkt og en rapport vi alle er stolte av.

Vi vil gjerne rette en stor takk til oppdragsgiveren vår Olav Gulbransen som alltid kommer med gode og konstruktive tilbakemeldinger til oss, samarbeidet med deg har vært uvurderlig og vi ønsker og takke for tillitten du har vist oss.

Vi ønsker også å takke vår veileder Rania El-Gazzar, for konstruktiv veiledning og gode tilbakemeldinger, du har vært til stor hjelp, tusen takk.

Houssam J. Melli

Henrik B. Hansen

Halvor U. Blindheim

Elin Handeland

Ketil Ustad

Kristian N. Singsaas

Sammendrag

Et kort sammendrag av rapporten og litt om hvordan vi har jobbet og samarbeidet vi har hatt med vår oppdragsgiver og veileder. I rapporten beskriver vi fremgangen og prosessene vi har vært gjennom under utviklingen av Nor IT sitt Professional Services Automation (PSA)-system, vi går også inn på ulike verktøy og metoder vi har tatt i bruk for gjennomføring av selve prosjektet. Vi har fått en del frihet når det kommer til hvordan systemet skal se ut og hvordan det skal operere av oppdragsgiver Olav. Dette har gitt oss muligheten til å utfordre kreativitet når det kommer til problemløsning og oppretting av ulike funksjoner og hvordan vi tenker at systemet skal utvikles. Vi har sammen med dette fulgt generelle retningslinjer og inputs som oppdragsgiver Olav skulle ha underveis.

Vi utvikler et PSA-system som har til hensikt å gjøre hverdagen for de ansatte ved Nor IT lettere. Systemet skal i teorien være et verktøy de ansatte kan benytte for å gjøre kundebehandlingsprosessene raskere og mer effektiv. For øyeblikket foregår mye av arbeidet manuelt gjennom epost og forskjellige portaler. I systemet vi designer ønsker vi å samle mye av funksjonaliteten Nor IT benytter seg av på en og samme plass, derav gjøre hverdagen mer effektiv for den ansatte.

Gjennom de tre årene vi har vært på Universitet i Sørøst-Norge (USN) har vi hatt et tett samarbeid, og den arbeidsmetodikken som vi har best erfaring med er Scrumban som er en blanding av arbeidsmetodene Scrum og Kanban (går litt mer i dybden på det senere i rapporten). Kort forklart lager vi en liste over oppgaver som trengs å gjøres deretter tar vi oppgaver fra en tavle (ClickUp), og når vi er ferdige med oppgaven markerer vi den som «løst» eller «ferdig».

Når det kommer til utfordringer og læring vi har hatt underveis i faget BOP3000, ønsker vi først å se litt på hvordan vi foretok risikovurderinger, for eksempel i henhold til Application Program Interface (API). Det viste seg å være lærerikt for vår del, ettersom uventede problemer dukket opp underveis i forbindelse med arbeid mot API-et til Fiken. Motivasjonen innad i gruppen har holdt seg på et godt nivå, men har hatt småperioder der den har duppet litt. Dette er noe vi tenker ikke er uvanlig og kan forekomme når man jobber på prosjekter som strekker seg over mange måneder. Dette er noe vi lærer av og tar med oss videre. En av tingene som har vært med på å gi oss motivasjon er møtene med veileder og oppdragsgiver. Vi føler vi har fått gode tilbakemeldinger fra begge to og det har gitt oss nytt pågangsmot og driv.

1 Innholdsfortegnelse

2 Introduksjon 1

3 Problemstilling 1

4 Bakgrunnslitteratur 2

5 Planlegging 3

5.1 Brief 3

5.2 Utviklingsmetode 6

5.3 Samhandling med oppdragsgiver og veileder 12

5.4 Teknologier 12

5.4.1 Utviklermiljø 13

5.4.2 Språk og rammeverk 14

5.4.3 Versjonskontroll 16

5.4.4 Andre 16

5.5 SWOT 16

5.6 Risikoplan 17

5.7 Estimert tidsforbruk for prosjektet 18

6 Analyse 19

6.1 MoSCoW 19

6.2 Funksjonelle krav 20

6.3 Ikke funksjonelle krav 21

6.4 Use cases 21

6.5 Dataflyt 23

6.6 Hosting 24

7 Design 25

7.1 Universell utforming av IKT 25

7.1.1 WCAG 2.0 25

7.1.2 WCAG 2.0-Standaren 26

7.2 Wireframes 26

8 Databasemodell 28

9 Sikkerhet 31

10 Systemarkitektur 34

11 System Infrastruktur 35

12 Kontinuerlig integrasjon og leveranse 38

13 Testing 39

13.1 Test metodikk og verktøy 39
13.2 Test Design 40
13.3 Praktisk gjennomføring av tester 41
14 Sideinnhold 43
15 Teknisk løsning 43
16 Fremtidige forbedringer og forandringer 45
17 Refleksjon 46
17.1 Generelle refleksjoner 46
17.2 Team refleksjon 48
17.3 General data protection regulation (GDPR) 49
18 Konklusjon 49
19 Referanser 50
20 VEDLEGG 54
20.1 Begrepsdefinisjoner 54
20.2 Gruppe kontrakt 59
20.3 Diagrammer 60
20.4 Utgåtte figurer og diagrammer 62
20.5 Brukerveiledning 67
20.6 User stories 73
20.7 Use Case diagram 74
20.8 Use Cases 75
20.9 Møtereferater 82

Figurliste

Figur 1: Viktige Datoer 5
Figur 2: Milepæler 5
Figur 3: Vår Scrumban metode 6
Figur 4: Kanban tavle 9
Figur 5: Mobile-First Design 11
Figur 6: Risikoplan 18
Figur 7: MoSCoW - Overordnede 20
Figur 8: MoSCoW - Funksjonelle krav 21
Figur 9: MoSCoW - Ikke funksjonelle krav 21
Figur 11: DFD level 2 - Sende ordre 24
Figur 10: DFD level 1 - Fakturere kunde 24
Figur 12: Wireframe - Hovedside Nor IT webapplikasjon 27
Figur 13: Wireframe - Legge inn ordrelinje på ticket 28
Figur 14: Databasemodell 29
Figur 15: Kardinalitet/Ordinalitet eksempel 30
Figur 16: Kardinalitet/Ordinalitet 30
Figur 17: Sikkerhet - Docker snyk scan 32
Figur 18: Sikkerhet - Docker snyk kjøreresultat 32
Figur 19: Sikkerhet - Docker snyk risikofunn 33
Figur 20: Sikkerhet - GitHub 33
Figur 21: Sikkerhet - GitHub actions secrets 34
Figur 22: Systemarkitektur diagram (detaljert) for Nor IT webapplikasjon – tredje utkast 35
Figur 23: Systemarkitektur diagram for Nor IT webapplikasjon – andre utkast 35
Figur 24: System infrastruktur 37
Figur 25: CI/CD diagram 38
Figur 26: Test design - Eksempel fra test design dokumentet 41
Figur 27: Tester - Kjøreresultater fra backend kodetester 41
Figur 28: Tester - Kjøreresultat fra route testing 41
Figur 29: Tester - Eksempel på testkode sakset fra forskjellige testfiler 41
Figur 30: Tester - Kjøreresultat fra en test i Postman fra en integrasjonstest 42
Figur 31: Tester - Snyk test av frontend 42
Figur 32: Tester - Tilgjengelighet og ytelsestest av frontend i Google Lighthouse 42
Figur 33: Sideinnhold - Nor IT webapplikasjon 43
Figur 34: DFD level 1 - Registrere kunde 60
Figur 35: DFD level 1 - Registere ny ticket/case 60
Figur 36: DFD level 2 - Timeregistrering på case 60
Figur 37: Mest populære integrated development environment 60
Figur 38: Mest populære web rammeverk 61
Figur 39: Mest populære databaser 61
Figur 40: Utgått systemarkitektur diagram for Not IT webapplikasjon - Første utkast 62
Figur 41: Utgåtte DFD diagrammer 63
Figur 42: Utgåtte databasemodeller 64
Figur 43: Utgåtte wireframes 66

Tabelliste

Tabell 1: Kommunikasjonsteknologier 11
Tabell 2: Estimert tidsforbruk 19
Tabell 3: Use Case - Ticketbehandling av ansatt 23
Tabell 4: WCAG 2.0 - Kravliste 26

2 Introduksjon

I denne rapporten skal vi presentere dere for det arbeidet og prosessen vi har jobbet med siden starten av januar, i faget BOP3000. I samarbeid mellom oss som gruppe, Universitet i Sørøst-Norge Campus Ringerike og Nor IT AS har vi hatt som oppgave å utvikle et Professional Service Automation system (PSA system) for oppdragsgiveren vår Olav. Gruppen vår har bestått av 6 studenter fra USN-Ringerike.

PSA-systemet som vi har utviklet har til hensikt å lette flere av arbeidsoppgavene som nå gjøres manuelt hos Nor IT, systemet skal også være oversiktlig og skalerbart. Under design perioden har vi prøvd å få med mange av funksjonene Olav kom med ønsker om under møtene vi har hatt sammen, og deretter har vi laget ett produkt som vi håper står til forventningene.

I rapporten og arbeidet foretatt har vi støtt på problemer her og der, men vi har også lært mye nytt. Små og store utfordringer har oppstått underveis og oppgaven vi fikk har vært stor i forhold til tid, men likevel ønsker vi å gjøre så godt vi kan for å tilfredsstille oppdragsgiver, og legge til rette for videre utvikling når vi i midten av juni skal overlevere prosjektet.

Noe av det viktigste vi tar med oss fra dette prosjektet at du trenger ikke å være en ekspert innen de forskjellige verktøyene du skal bruke, selv om du ikke har brukt dem før. God kommunikasjon, viljen og ønsket om å lære, tilegne seg kunnskap og samarbeid er gjerne oppskriften til en god oppgave. Det har vi kjent på flere ganger under vår tilstedeværelse hos USN, hvor gruppeoppgaver har vært en sentral del av vårt studium.

3 Problemstilling

> *Hvordan lage en brukervennlig PSA applikasjon som ansatte hos Nor IT kan bruke med utgangspunkt i at de ekspanderer og ønsker å automatisere funksjonalitet.*

Oppgaven går ut på å lage en PSA-applikasjon som skal automatisere deler av de administrative prosessene i firmaet Nor IT. I dag er det flere av prosessene i firmaet som gjøres manuelt, noe som tar opp mye tid og ressurser og øker potensialet for feil og tapt inntekt. Grunnet den mulige kompleksiteten og API-first fremgangsmåten i oppgaven vil vi i utgangspunktet lage en modulær applikasjon hvor de mest grunnleggende funksjonalitetene er på plass, slik at man kan utvide applikasjonen ved en senere anledning.

Vår bachelorgruppe skal i denne oppgaven utvikle et system hvor de ansatte hos Nor IT har mulighet til å registrere tidsbruk, både når de jobber med ulike caser på kontoret, men også når de er ute hos kunder. Systemet vil bestå av ticketing, hvor det i tillegg til tidsregistrering også skal være mulig å legge inn ordrelinjer bestående av varer og tjenester. Denne ordrelinjen skal være mulig å sende til fakturering i Fiken. Videre vil vi jobbe med å få på plass generell kundesupport samt oppfølging og administrerings muligheter med tanke på kundeforespørsler.

I starten av oppgaven så vi for oss kommunikasjonen med kundene til Nor IT skulle foregå via en ticket og at vi så på selve ticketen som en kommunikasjonsportal mellom kunde og bedrift. Kundene vil kunne logge seg inn i applikasjonen gjennom en forhåndsregistrert bruker, det skal også lages en simple mail transfer protocol (SMTP)-integrasjon for dialog med kunder, Brønnøysundregisteret og regnskapsapplikasjon.

Når vi nå har kommet lengre ut i prosessen ser vi at dette er unødvendig etter et møte vi hadde med Olav (se møtereferat.), og vi vil i denne omgang utforme et system hvor det ikke vil være noen innlogging for kundene, men at vi skal innlemme mailsystemet til Nor IT inn i ticketen, slik at kundene til Nor IT ikke trenger å logge seg inn på noe annet system enn å benytte sin e-post slik som de vant til. For Nor IT vil dermed både kommunikasjon og nødvendig informasjon samles på et sted, og de vil unngå og måtte bytte mellom ulike systemer for å registrer tid, legge til eller fjerne produkter, gjøre seg notater for hver enkelt kunde og i tillegg ha e-post i ulike programmer.

Begrepsdefinisjon

Ticketing blir et sentralt begrep i våres bachelor prosjekt. Ticketing eller en ticket vil være en del av et system som bidrar til at bedriften kan ha all interaksjon med kundene sin på et sted. I en ticket kan bedriften legge inn notater ang kundeforhold, varer og tjenester kunden ønsker å kjøpe, hva kunden trenger av support og eventuelle tjenester (sysaid.com, u.å.).

4 Bakgrunnslitteratur

Vi har i stor grad benyttet oss av kunnskap fra emnet systemutvikling. I boken fra dette emnet Beginning Software Engineering av R. Stephens, fant vi både relevant og god litteratur om hvordan man utvikler etter System Development Life Cycle (SDLC) som opprinnelig består av seks ulike faser, hvorpå den siste av disse vil ikke bli benyttet under dette prosjektet, med

bakgrunn i at vi skal ikke stå for drift og vedlikehold av systemet etter leveringen av dette Bachelor prosjektet. Dermed blir vårt prosjekt bestående av de fem første fasene, planlegging, analyse, design, implementering og testing (Stephens, 2015, s.276-277). Videre har vi tilegnet oss viktig kode kunnskap fra fagene grunnleggende programmering 1 og 2 hvor Python kunnskapene fra fagene har kommet godt med i implementeringen. I fagene brukte vi boken Starting out with Python av T. Gaddis hvor vi har lært og funnet relevant informasjon. Vi har også hatt nytte av emnene objektorientert programmering når det kommer til den delen av systemet hvor vi har benyttet Java som programmeringsspråk. I femte semester hadde vi faget SEL, under dette prosjektet tilegnet vi oss en dypere forståelse rundt det og kunne reflektere over ulike emner, vi fikk også god trening i det og strukturere både arbeidet, innhenting av relevant informasjon og produsere en rapport på en strukturert og ryddig måte. Vi har også satt oss inn i tidligere bacheloroppgaver og sett på struktur og hentet inspirasjon fra de beste bacheloroppgavene fra de siste 4 årene. Når det kommer til databasemodellering og oppsett av databaser og hvordan gjøre dette på en ryddig og fornuftig måte har vi tatt med oss kunnskap fra emnene Database 1 og 2, hvor vi også har benyttet oss av faglitteraturen fra bøkene Databasesystemer av B. Kristoffersen og Database System Concepts av A. Silberschatz, H. F. Korth og S. Sudarshan.

5 Planlegging

I dette kapitlet skal vi forta en gjennomgang av hvordan vi planlagte å utføre oppgaven vi fikk fra oppdragsgiveren vår. Det vil være litt generell informasjon om prosessen og oppgaven. Litt om oss som gruppe vil også forekomme. Vi kommer til å gå gjennom litt teori og teknologier vi har benyttet oss av, diagrammer og analyser med hensyn til oppgaven og gruppen.

5.1 Brief

Om prosjektet

Prosjektet Bach-PSA Nor IT er et samarbeid mellom MH_2EK_2 (studentgruppens kallenavn), USN og Nor IT og utgjør bacheloroppgaven for MH_2EK_2. Prosjektet skal resulterer i leveransen av et PSA som letter arbeidsmengden hos ansatte i Nor IT i forbindelse med kundekommunikasjon og fakturering. Systemet skal også forenkle og forkorte kunder eller mulige kunder av Nor ITs kontaktprosess. Det er viktig at systemet legger vekt på fremtidige tilpasninger og videreutvikling.

Leveransebeskrivelse

Prosjektet endeprodukt skal være funksjonelt og inneholde de elementer som er definert under kolonnene M og S i MoSCoW-diagrammet. Det bør være tilrettelagt for eller ligge klart i en skytjeneste som Microsoft Azure. Det bør inneholde en autentiseringstjeneste ala Azure AD eller liknende.

Objektiver og utfordringer

Systemet bør utvikles med Mobile first vinkling på grensesnitt og API first som vinkling på datatrafikk, dataprosessering og databehandling. Mobile first er prosjektgruppen godt kjent med som metodikk fra universitetsstudier, mens API utvikling er et element vi er nødt til å tilegne oss. Prosjektgruppen ønsker å oppnå bemerkelsen for beste bacheloroppgave for våren 2022.

Suksesskriterier

For å sørge for at dette prosjektet skal lykkes har vi identifisert fire hovedpunkter vi skal etterleve. Disse punktene stammer fra forrige prosjekts retrospektive vurdering.

Åpen og jevnlig kommunikasjon Internt i gruppen

Vi må gjøre daglige arbeidsmøter på de dagene vi arbeider, og ha en retrospektive/review når vi når en av milepælene vi har satt oss.

Eksterne interessehavere

Prosjektet krever 2 obligatoriske møter mellom prosjektgruppen, veileder hos USN og oppdragsgiver (Nor IT) For at vi skal lykkes er det viktig for oss med et tettere samarbeid med oppdragsgiver enn kun to møter. En til to ganger i måneden vil være ideelt.

Jevn og organisert arbeidsflyt

Kontinuerlige samtaler om gjøremål og arbeidstider er elementært for at vi som gruppe skal lykkes med prosjektet.

Struktur

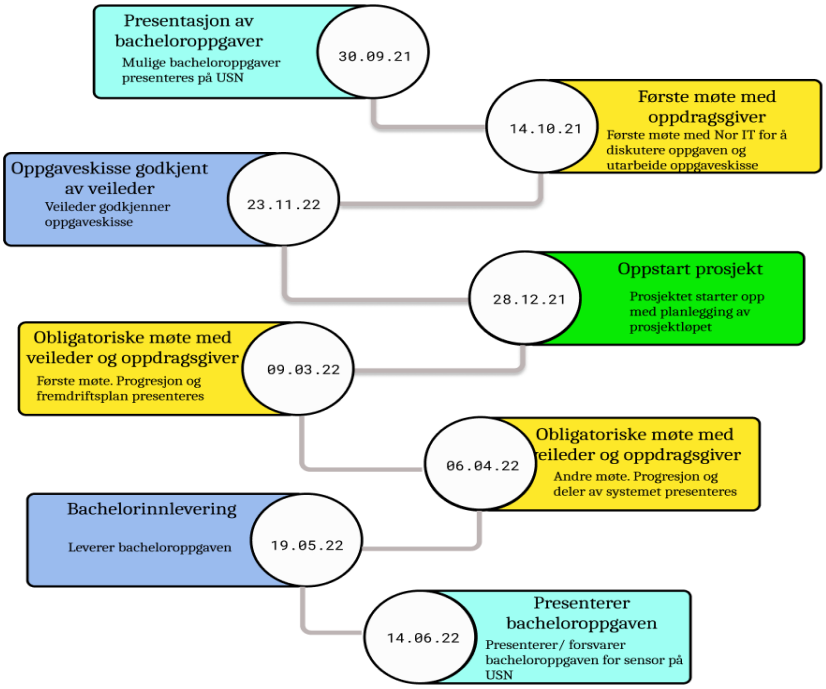
Det er viktig for oss at vi er nøye med planlegging og følger planene våre. Det igjen betyr at vi må følge livssyklusplanene vi setter for oss, uten å hoppe for mye frem og tilbake. Slik ungår vi å møte på problematikk vi har møtt på i tidligere prosjekter.

Innsikt i og vilje til å gjøre det lille ekstra

Dette prosjektet har fra USNs side en estimert tidsbruk på 375 timer per person i prosjektgruppen. Våre egne estimater legger oss på en forventet arbeidsmengde på ca. 410

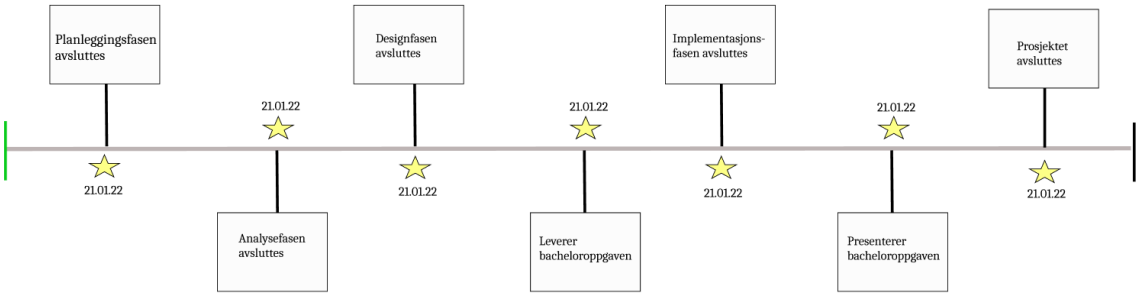
timer. Det betyr at gruppen som helhet og på individnivå må være villige til å jobbe utover det som forventes fra universitetets hold.

Viktige Datoer



Figur 1: Viktige Datoer

Milepæler



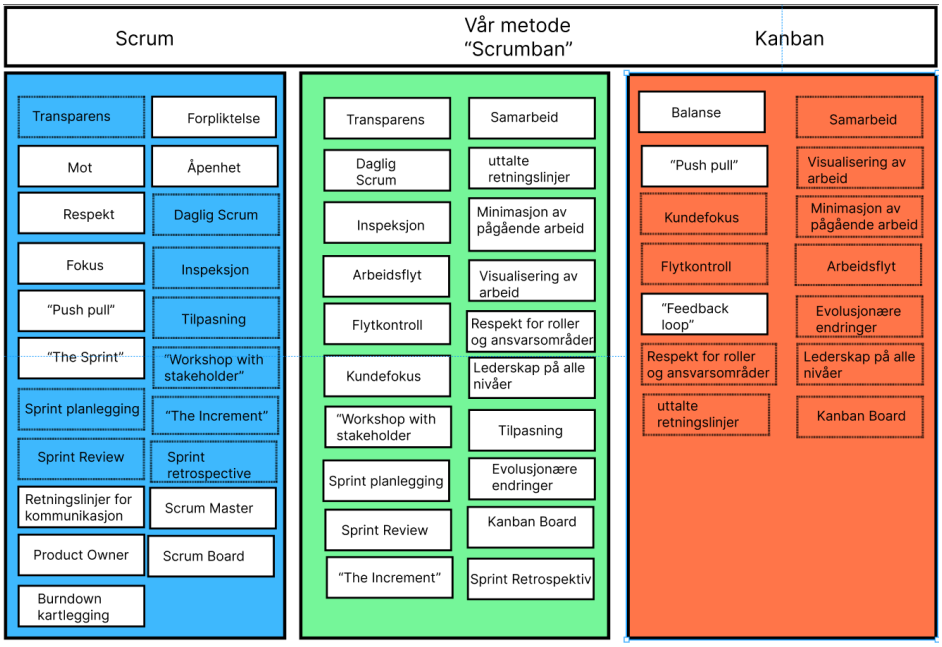
Figur 2: Milepæler

Kommunikasjonsplan

Internt i prosjektgruppen er det avtalt «stand up's» kl. 10:00 hver arbeidsdag, som er tirsdager og onsdager. Vi har avtalt bi-ukentlige møter med oppdragsgiver (Nor IT) annenhver tirsdag på slutten av arbeidsdagen. Det blir avsatt to tirsdager til møte mellom veileder, prosjektgruppe og oppdragsgiver samlet.

5.2 Utviklingsmetode

For å kunne lykkes i et prosjekt er det viktig å legge til rette for god prosjektførsel ved å arbeide etter et klart rammeverk. Vi som gruppe har fra tidligere prosjekter erfart at Scrumban passer for oss, og valgte derfor å arbeide etter Scrumban modellen i dette prosjektet også. Kort fortalt er Scrumban en hybrid arbeidsmetode som plukker ut ulike elementer fra begge arbeidsmetodene Kanban og Scrum (Owcarz, 2021). Figuren under avsnittet (figur 3) viser noen av de viktigste kjerneprinsipper, retningslinjer og praksiser i Scrum og Kanban og hvilke av disse vi har valgt å ta fra hvert av de to rammeverkene. Figuren fungerer på den måten at de elementene i den blå eller oransje kolonnen som har transparent bakgrunn og stiplet ramme vil være å finne i vår tilpassede versjon av Scrumban, mens de elementene med hvit bakgrunn og heltrukken ramme er elementer ved de respektive rammeverkene vi ikke har valgt å ta med oss. Figur 3 viser i midtre kolonne de valgte elementene med sin originale tekst og betydning, men i noen tilfeller har vi tilnærmet oss disse på en noe tilpasset måte.



Figur 3: Vår Scrumban metode

Våre «sprints»

Der hvor Scrum har ganske klare definerte retningslinjer for hvordan man behandler en sprint. Har vi valgt å se litt annerledes på hvordan vi organiserer fasene i vårt prosjekt. Det har vi gjort fordi vi så det ville bli vanskelig å finne en god måte å organisere tradisjonelle Scrum sprints på som var gjennomførbare for vår del. Vi hadde ikke oversikt over

kompleksitet og nødvendig tidsbruk i forbindelse med obligatoriske arbeidskrav i andre emner ved universitetet og dette var hovedårsak til at vi gjorde de valgene vi gjorde i denne forbindelsen. For å fordele vår tid etter beste evne, og samtidig ha nok milepæler og feire valgte vi å dele prosjektet opp i faser. Vi valgte å følge rammeverket for livssyklusen for programvareutvikling for navngivning av våre faser. Denne navngivningen har hjulpet oss å holde styr på hvilke typer oppgaver som må gjøres før vi kan gå videre til neste fase og i flere tilfeller stoppet oss fra å hoppe over et steg og gå rett i gang med kodingen av systemet. Vi har hatt fasene planlegging, analyse, design og implementering. Livssyklusfasen testing og integrasjon har hos oss inngått i implementeringsfasen da vi har ansett det som nødvendig å teste fortløpende og sørge for sømløs integrasjonen mellom systemets forskjellige komponenter. I planleggingsfasen gikk vi grundig gjennom oppgaven og planla for og estimerte tidsbruk i hver av de andre fasene og satt opp en tidsplan for hver av disse. Samtidig som det ble gjort satt vi sammen en backlogg for hver av fasene slik at vi ved behov hadde planlagt for oppgaver dersom vi så at vi enten kunne, eller fikk behov for å splitte teamet og la noen starte arbeid i neste fase mens vi ventet på avklaring på noe arbeid i den «aktive» fasen. Avslutningsdatoen for fasene ble definert som en milepæl sammen med enkelte andre større eller viktige arbeidsoppgaver som f.eks. første obligatoriske møte mellom alle gruppen, veileder og Nor IT. Ved avslutningen av hver fase har vi hatt et møte hvor vi har kjørt vår variant av en sprint review, en sprint retrospektiv og «the increment». Under dette møtet har vi snakket, gått gjennom alle oppgaver gjort for denne fasen og i plenum godkjent eller gitt denne en status som må gjøres og med høy prioritet. Deretter har vi frigjort alle klare og godkjente oppgaver for leveranse ved f.eks. å innlemme arbeidet i rapporten. Til sist i dette møtet har vi diskutert hva som har fungert og hva som ikke har fungert og brukt tilbakemeldingene her til å tilpasse neste fase. Vi hadde som regel at vi hadde dette møtet som siste hendelse for arbeidet med prosjektet i uken vi var i. Da fikk vi sørget for at vi startet med ny fase i ny uke og med nytt pågangsmot.

Under kommer litt mer om hvordan vi har arbeidet med tanke på de elementene vi har valgt å benytte i vår Scrumban.

Grunnleggende prinsipper

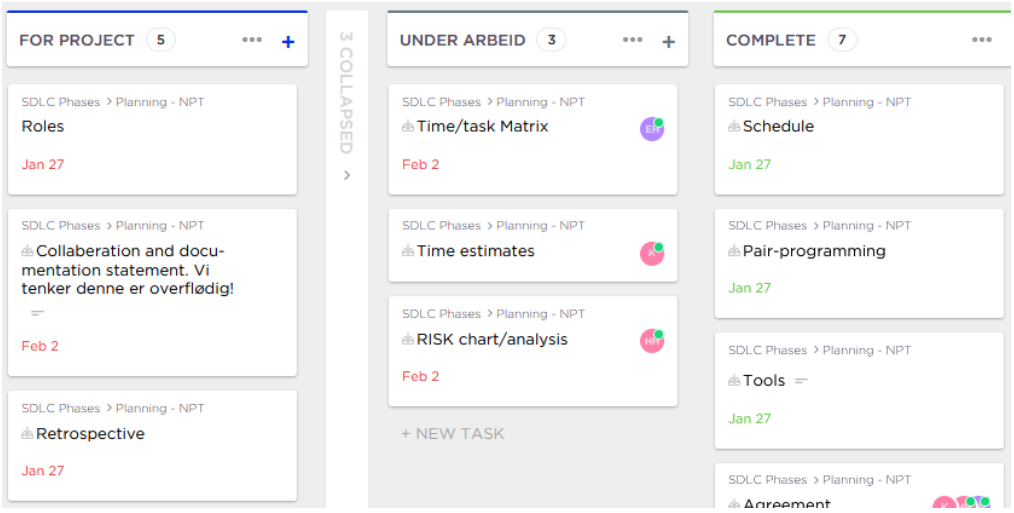
På Scrum.org kan man lese om de tre empiriske pilarene for Scrum. Disse er transparens, inspeksjon og tilpasning (Doshi, 2016). Det er disse tre som også danner grunnlag for vår prosjektførsel. Vi bruker transparens i alle våre oppgaver og kommunikasjon. I alt fra å dele hvilke oppgaver vi arbeider med, hvordan vi opplever samarbeidet, vansker vi møter på

personlig og i prosjektet både internt i gruppen og med de andre interessentene som Nor IT og vår veileder. Inspeksjon gjør vi ved at vi har som prinsipp at den interne prosjektgruppen, det vil si alle foruten Nor IT og veileder, er med på å dele pågående arbeid og også godkjenner og avstemmer disse ved enden av en av våre utviklingsfaser. For å kunne etterleve prinsippet om tilpasning har vi arbeidet etter en tilpasset versjon av Kanban evolusjonære forandringer valgte vi bort å jobbe målstyrt og kun arbeide samarbeidende. Vi har hatt jevnlig kommunikasjon med Nor IT og som gruppe for å avdekke hvor vi har behov for forandringer enten i prosess eller i produkt. På denne måten har vi utviklet oss kontinuerlig gjennom hele prosjektet.

Flytkontroll

En del av prinsippene og praksisen for flytkontroll er nevnt i avsnittet «vår sprint» men vi har gjennomført en del flere elementer også. Disse gjør seg gjeldende under paraplyen «arbeidsflyt». Her ligger elementer som å minimalisere pågående arbeid, visualisering og uttalte retningslinjer. Visualiseringen av arbeidet, både hva som må gjøres og hva som er gjort gjør det enklere å holde oversikt over hvordan arbeidet flyter. Måten å visualisere arbeidsoppgaver på har vi tatt fra Kanban hvor en stor del av hvordan det jobbes med visualisering av hva som må gjøres på en Kanban-tavle som vist i bildet under dette avsnittet (Hamilton, 2021). En Kanban-tavle fungere slik at man legger arbeidsoppgaver i form av kort eller rubrikker i en kolonne på en tavle. Denne tavlen kan være enten digital eller fysisk. Vi har valgt å benytte oss av ClickUp for denne funksjonen. Når en oppgave ligger i kolonnen for ledige oppgaver på tavlen vil den være åpen for at et medlem av utviklingsgruppen kan ta eierskap i denne, og flytte denne oppgaven for oppgaver under arbeid. Når oppgaven er ferdigstilt og klar for kontroll flyttes den til kolonnen for ferdige oppgaver. Deretter vil oppgaven kontrolleres, godkjennes og leveres, da vil denne oppgaven arkiveres. Vi har hatt som klare uttalte retningslinjer at man tar eierskap til et arbeid ved å skrive sitt eget navn på kortet når man begynner på en oppgave og flytter dette kortet i samsvar med de retningslinjer Kanban setter for flyt på tavlen. I tillegg har vi uttalt at dersom en ny oppgave, eller tilleggsoppgave dukker opp under arbeid, er det oppgavens eier som har ansvar for å oppdatere backlogg for å dekke denne. Grunnen til at vi ønsket at man skulle angi eierskap er fordi vi da kan vi enklere kan gå gjennom oppgaver når alt skal presenteres og godkjennes for leveranse og vi slipper å gjette hvem som har gjort hva. I tillegg hjelper det oss å holde kontroll på et av Kanban' ufravikelige krav om å minimalisere pågående arbeid. Vi har uttalt et krav om at hver og en av oss kun kan ha et pågående arbeid, med mindre man

samarbeider med noen andre om en oppgave og da deler eierskap til den oppgaven. Vi har også valgt å ha godt kundefokus og organisere workshops med kunde, som i vårt tilfelle er Nor IT. Jevnlig kommunikasjon og samarbeid stopper oss fra å gjøre en masse unødvendig ekstra arbeid og at vi levere på det vi skal. Ved å kjøre workshops har vi fått avdekket kravene til systemet og også fått forkastet noen av de underveis. Det har spart oss fra å skulle tenke ut løsninger på egenhånd, sende epost eller bare produsere, for så å finne ut at vi ikke var helt på rett spor (Meiling, 2019).



Figur 4: Kanban tavle

Generelt om samarbeid

Som gruppe har vi valgt å kjøre vår versjon av en «daily Scrum» eller morgenmøte. Det har vi gjort som første hendelse hver arbeidsdag. Her har vi gått gjennom agenda for dagen, dersom noe spesifikt sto på planen f.eks. møte med kunde eller veileder. Så har hver enkelt deltager i møtet gått gjennom sin plan for dagen, med hvilke oppgaver man skal gjøre og her har vi hatt som uttalte retningslinje at vi skal være så konkrete som mulig. Dersom man opplever noe som vanskelig eller sitter fast på en oppgave er det også noe som ble tatt opp i dette møtet. Denne tilnærming har vært med på å sørge for flytkontroll, nevnt i paragrafen over, og har sørget for at vi har god daglig arbeidsflyt. En bi-effekt av dette møtet har vært at vi har avdekket at noen har glemt å ta eierskap eller flytte et Kanban-tavle kort og flere har påbegynt arbeid med det samme. Da har vi fått korrigert dette før for store ressurser har blitt kastet bort. Der hvor Scrum klart definerer påkrevde roller i en Scrum Master og en Product Owner og et Developer Team har vi valgt å se på roller i en litt annen setting. Vi har hatt klare definerte roller som i Scrum, men har valgt å kombinere dette med Kanban' prinsipp om lederskap i alle roller. Det har gjort at vi har delt opp ansvaret slik at nær sagt alle har hatt

en rolle hvor lederskap er en naturlig del av rollen. Vi har hatt noen ansvarlige for frontend, backend, prosjektførsel og rapport, hvor andre igjen har hatt medansvar. Vi har også hatt som uttalte retningslinje at til tross for at noen står med «hovedansvar» for en del av prosjektet er vi alle medansvarlige i alle aspektene av den totale oppgaven. Når spørsmål om hvordan ting bør løses har disse blitt rettet til den som har hatt hovedansvaret for den delen av oppgaven som blir berør av spørsmålet. Med den fremgangsmåten har vi sørget for at rollene ikke kun har vært en tittel, men hatt praktisk og faktisk betydning. Vi har også hatt klare uttalte retningslinjer for hvordan vi kommuniserer med hverandre og når vi kan forvente svar. Møter hvor alle deltar har vi avholdt på teams fordi det gjør det lettere å komme i kontakt med hverandre om noen skulle ha utelatt fra å møte til avtalt tid, samt at vi har alle felles dokumenter lagret der. Discord har vi brukt som kommunikasjonsplattform når folk er aktive med arbeid slik at man har vært tilgjengelig til enhver tid og man enkelt kan hoppe mellom kanaler dersom man har behov for å snakke med flere uten å forstyrre alle. For direktemeldinger i tekst har vi opprettet en gruppe på Facebook Messenger hvor alle meldinger er åpne slik at alle kan lese. Vi har hatt som uttalte retningslinje at et spørsmål skal besvares av den det rettes til samme dag, med mindre kommer utenfor definert arbeidstid 09:00 – 15:00 da skal det besvares senest neste dag.

Beskrivelse av kommunikasjonsteknologier

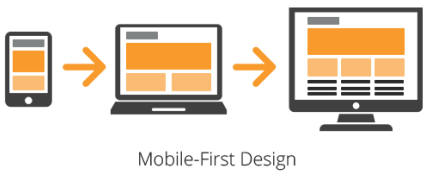
	Facebook Messenger: Dette er hovedprogrammet vi bruker for å holde kontakt og kontakte hverandre med beskjeder. Messenger har også en fin app for mobil som gjør det lettere å nå hverandre når og hvor som helst. Det fungerer i praktisk likt som SMS på mobil hvor vi har laget en gruppe hvor vi chatter med hverandre.
Microsoft Teams	**Microsoft Teams**: Når vi har møter bruker vi Microsoft teams. Det har også blitt laget en gruppe på Microsoft teams hvor vi deler blant annet filer og bilder med hverandre til bruk i prosjektet. Flere kan også jobbe på samme fil gjennom teams noe som blir anvendt når vi skriver denne rapporten og alt blir oppdatert real-time. Dvs. at endringer som en i gruppen gjør vil bli synliggjort for alle andre samtidig

Discord	**Discord**: Dette programmet fungerer også som et kommunikasjonsprogram. Her har vi flere tekst- og talekanaler hvor vi fordeler oss på når vi setter oss ned for å jobbe etter Scrum-møtene på morgningen. Programmet er kjekt ved at det tillater oss å selv lage og sette opp det vi ønsker av disse kanalene for å holde kommunikasjon/informasjon relevant og lett å finne. Det er også mulig for oss å «streame» eller dele det vi ser på vår egen skjerm for de andre. Ofte når vi jobber på samme diagram og ikke har mulighet til å møte fysisk er denne funksjonen god å ha, eller hvis man bare ønsker å dele noe man jobber med og trenger innvendinger

Tabell 1: Kommunikasjonsteknologier

Utviklerstrategier

API-First metoden er en relativt ny utviklermetode hvor utviklerne koder konsistent og gjenbrukbare API-er, dette er viktig om en skal følge API-First metoden. I dag bruker de aller fleste bedrifter API-er og det er blitt en relativt fundamenter implementasjon. I API-First blir API-et hovedfokuset når det kommer til implementasjon. All funksjonalitet som går gjennom applikasjonen skal kunne nås gjennom API-ete, denne strategien forsikrer at prosjektet blir utviklet og utformet etter API-ets design. Fordelene ved å utvikle i henhold til API-First er i korte trekk effektivitet, skalerbarhet, gjenbruk, utvikling på tvers av plattformer og utformer bedre utvikler erfaringer (Bilal, 2021).



Figur 5: Mobile-First Design 2021, av Design-Lance http://design-lance.com/tag/mobilefirst/

Mobile first design handler om hvordan man planlegger og utvikler nettsider på en måte hvor man legger til rette for bruk på mobile enheter og har dette som utgangspunkt og primærflate. Gjennom emnene WEB1100 – Webutvikling og HCI samt APP2000 Applikasjonsutvikling

for web har vi lært om dreiningen mot mobile flater i markedet og mobile first som etablerte strategi bransjen. Vi valgte å følge denne strategien i vår design og planlegging av systemet vi produserer, men valgte å se bort ifra den i koden. Det valget ble tatt i samråd med oppdragsgiver og med tanke på tid og kompleksitet. Det er likevel lagt til rette i koden for at applikasjonene skalerer opp, og ikke ned. Men vi har bare kodet den oppskalerte biten av den.

5.3 Samhandling med oppdragsgiver og veileder

I dette prosjektet har vi et ønske om å ha et tett samarbeid med oppdragsgiver, Olav Guldbrandsen hos Nor IT AS. Vi har avtalt og ha jevnlige møter med han på treklyngen Follum Nord med møte intervall på hver 14 dag. Dette kan bli endret underveis i prosjektet, hvis vi skulle se at det ikke vil være noe behov for å møtes så ofte. Vi ser ingen grunn til å ha møter kun for å ha møter, og ønsker derfor at vi skal ha en agenda før hvert møte. Olav har også formidlet at vi kan ta kontakt med han utenom disse fastsatte møtetidspunktene, om det skulle være noe vi trenger svar på.

Vi har planlagt og avtalt to ulike datoer for å ha møte med veileder Rania El-Gazzar, og Olav sammen. Dette møte vil finne sted på USN Campus Ringerike. Det første fellesmøte vil finne sted onsdag 09. Mars, mens det andre møte vil bli holdt etter påske, fredag 22. April. Datoene er valgt med bakgrunn i hvor i prosjektet vi vil være. Når vi avholder det første møte skulle vi egentlig vært ferdig med designfasen. Nå viste det seg at vi hadde avsatt for kort tid til denne fasen, men vi hadde uansett relevante ting og vise frem til Olav og Rania, slik at de kunne komme med tilbakemeldinger på arbeidet vi hadde gjort så langt. Når det kommer til det andre møte har vi valgt å legge det til fasen hvor vi jobber med implementasjonen. Slik at vi da kan ha noe nytt å vise frem og ikke minst få tilbakemeldinger på fra både veileder og oppdragsgiver.

Vi tar møtenotater for alle møter slik at ikke noe informasjon blir utlatt eller glemt av oss etterpå, vi har komprimert dem og skrevet utdrag fra hvert møte som du finner under vedlegg.

5.4 Teknologier

“To many programmers, development is the heart of software engineering. It’s where fingers hit the keyboard and churn out the actual program code of the system. Without development, there is no application” - Rod Stephens (Stephens, 2015, s. 143).

Rod Stephens forteller at det er viktig med god programvare for et godt integrert utviklingsmiljø. Programmene man bruker trenger ikke nødvendigvis å være de mest fancy,

men at det finnes masse gode programmer der ute med mulighet for ekstra funksjonalitet hvis man ønsker å bruke penger på å oppgradere til «professional» eller for eksempel «ultimate» utgaver. Rod Stephens nevner også at "A good source code management system" som betyr å kunne se historikk på koden man lager og se hvilke endringer som blir gjort er viktig for å kunne gå tilbake og se hvor en eventuell feil skjedde. Det gjør det også lettere når det er flere personer som koder på samme program ved bruk av god «source code control programs» (Stephens, 2015, s. 146-147). Med dette som basisgrunnlag og tatt i betraktning velger vi ut teknologier eller programvare for vår utvikling som vi har god erfaring med og vet har gode innebygde funksjoner, samt mulighet for utvidelse og integrering til det vi måtte trenge.

5.4.1 Utviklermiljø

Det finnes mange gode utviklingsmiljøer, og det går i stor grad på personlige preferanser på hvilket man foretrekker å bruke. De aller fleste av de store som Eclipse, Intellij, Visual studio code for å nevne noen har veldig god innebygd versjonskontrollmulighet med plug-ins eller utvidelser som gir dem funksjonalitet på det du måtte ønske. Ved å betale og oppgradere til en premium versjon vil du også kunne få tilgang til funksjonalitet som er nyttig i store prosjekter, men ikke nødvendig hvis du koder for deg selv (Stephens, 2015, s. 146).

Visual studio code er en kode editor laget av og utviklet av Microsoft. Her kan man skrive kode i de fleste programmeringsspråkene. For å nevne noen: Java, javascript, python, C++, Hypertext Preprocessor (PHP), Cascading Style Sheets (CSS). Dette programmet er også definitivt det mest brukte som utviklingsmiljø innenfor systemutvikling (som vist i vedlegg under kapitelet diagrammer).

De som har utviklet Visual studio code har også lagt til rette for gode muligheter med utvidelser, debugging og personalisering. Visual studio code er også støttet som programvare på både windows, macOS og Linux, samt et stort fokus på rask responstid og støtte til de fleste programmeringsspråk har blitt lagt vekt på (Visualstudio, 2021).

Likt som Visual Studio Code er også PyCharm en kode editor. Men PyCharm er vesentlig forskjellig ved at den ikke er «universell» men er en dedikert Integrated Development Environment (IDE) for kodespråket Python. Den støtter også vesentlige verktøy og språk for Python utviklere som HyperText Markup Language (HTML), CSS, JavaScript og Extensible Markup Language (XML) (JetBrains, 2022). PyCharm er utviklet av det Tsjekkiske programutviklingsselskapet JetBrains (https://www.jetbrains.com/) og er støttet på Windows, macOS og Linux. Vi bruker PyCharm til å utvikle API-et fordi det tilbyr gode verktøy og «code completion».

5.4.2 Språk og rammeverk

Html. CSS. Javascript. JavaScript XML (JSX). Python. PostgreSQL(sql-dialekt)

For å bygge brukergrensesnittet som retter seg mot de ansatte hos oppdragsgiver har vi valgt oss React.js som rammeverk/bibliotek. Vi valgte React.js fordi det er den mest utbredte teknologien for utvikling av webapplikasjoner (som vist i vedlegg under kapitelet diagrammer).

Ikke bare er React.js den mest populære teknologien, men den tilbyr også en helt annen form for smidighet når det kommer til bruk av tilleggsmoduler og biblioteker enn hva rammeverk som Vue.js og Angular.js kan tilby. React.js applikasjoner kan utvikles både i JavaScript og TypeScript. Vi har valgt JavaScript da det er dette vi er best kjent med og da slipper unna TypeScripts forholdsvis høye læringskurve. JavaScript er også den mest utbredte teknologien som gjør det enklere for oss og finne god veiledning til de utfordringer vi vil støte på. Vi slipper også å ta høyde for kompilering av koden slik vi måtte ha gjort med TypeScript.

FastAPI er et rammeverk for Python som benyttes til å bygge et API. For vår del har vi valgt å bygge et Representational State Transfer (REST) API som vil ligge som en middleware mellom våre frontend-tjenester, våre eksterne datakilder og vår database. Vi har valgt oss

REST som arkitektur for vår API fordi det er den de-facto standarden for moderne API utvikling (Spring, u.å.).

Grunnlaget for å velge akkurat FastAPI som rammeverk for utviklingen er flere. Rammeverket benytter som sagt Python som utviklerspråk og dette er et språk flere i gruppen kjenner godt til. I tillegg er veien fra oppstart av til man kan begynne å servere data betraktelig kortere enn andre rammeverk i da FastAPI som rammeverk er langt mindre omstendelig. FastAPI har også typehinting som standard praksis i sine funksjoner og benyttelse av objektorientert klasser som sørger for at den utviklede koden i langt større grad blir selvdokumenterende. FastAPI sørger også automatisk for dokumentasjonen av selve API-løsningen slik at vi slipper å bruke tid på det.

Før vi valgte oss FastAPI undersøkte vi også om Spring kunne være aktuelt som rammeverk for vår API, men fant ut at dette ikke var aktuelt både av kompleksitetsgrunner, men også med henhold i nylige sikkerhetsavvik i et større Java logging-bibliotek og da tenker vi spesifikt på log4jrce exploiten.

Til tross for vår valgte fremgangsmåte med API-first for data og databehandling, trenger vi et sted å persistere data. Til dette har vi valgt PostgreSQL. Postgre er en SQL-dialekt og en teknologi vi er kjent med fra utdanningsløpet. Årsaken til at valget falt på Postgre over f.eks. MySQL ligger i teknologienes arkitektur. Postgre har en objekt-relasjonell arkitektur som legger bedre til rette for oppskalering, flere og mer komplekse datatyper, samt muligheten for opprettelse av brukerstyrte datatyper. Dette er funksjoner som gjør det lettere for oss å tilpasse datamodellen vår use case.

Postgre er som vist i vedlegg en veletablert teknologi i bransjen og kan derfor tilby god dokumentasjon og brukerveiledning fra mange diverse kilder.

Valget av Postgre betyr at vi også har valgt en relasjonell datamodell for applikasjonen vår. Det er et valg vi har tatt fordi dataen vi skal lagre er klassisk relasjonell type. Vi skal lagre forholdsvis små entiteter av forskjellige klasser som gjør seg godt fordelt på tabeller som viser til hverandre som f.eks. data om en bruker, data om en ordrelinje og en faktura.

Relasjonelle databaser er også noe vi har mer erfaring med enn objekt orienterte databaser som gjerne brukes for mer komplekse typer data.

5.4.3 Versjonskontroll

Versjonskontroll eller «source code control programs» brukes for å sikre oss at det er mulig å kunne se historikk på kode som blir skrevet, samt ha sikkerhetskopier tilgjengelig tilfelle noe skulle gå galt. Det blir brukt som et sikkerhetsnett slik at filer og endringer ikke skal gå tapt, og at man kan sammenligne med tidligere versjoner. Som vårt versjonskontrollsystem benytter vi oss av teknologien utviklet av GIT (https://git-scm.com/). Det finnes flere alternativer til GIT som Azure DevOps Server, Helix Core og Mercurial for å nevne noen.

Koden vår hoster vi på GitHub. Vi benytter oss av enkeltstående repositories for hver av de større systemkomponentene av det helhetlige systemet vi utvikler. GitHub tillater oss å samarbeide i sanntid på koden og gir oss mulighet til å sende koden til server på en enkel måte når den tid kommer. Vi har også muligheten til å legge inn automatiske tester som kjøres på hvert bidrag til kodebasen samt kjøre automatiske oppdateringer av koden på serverne etter CI/CD metodikk.

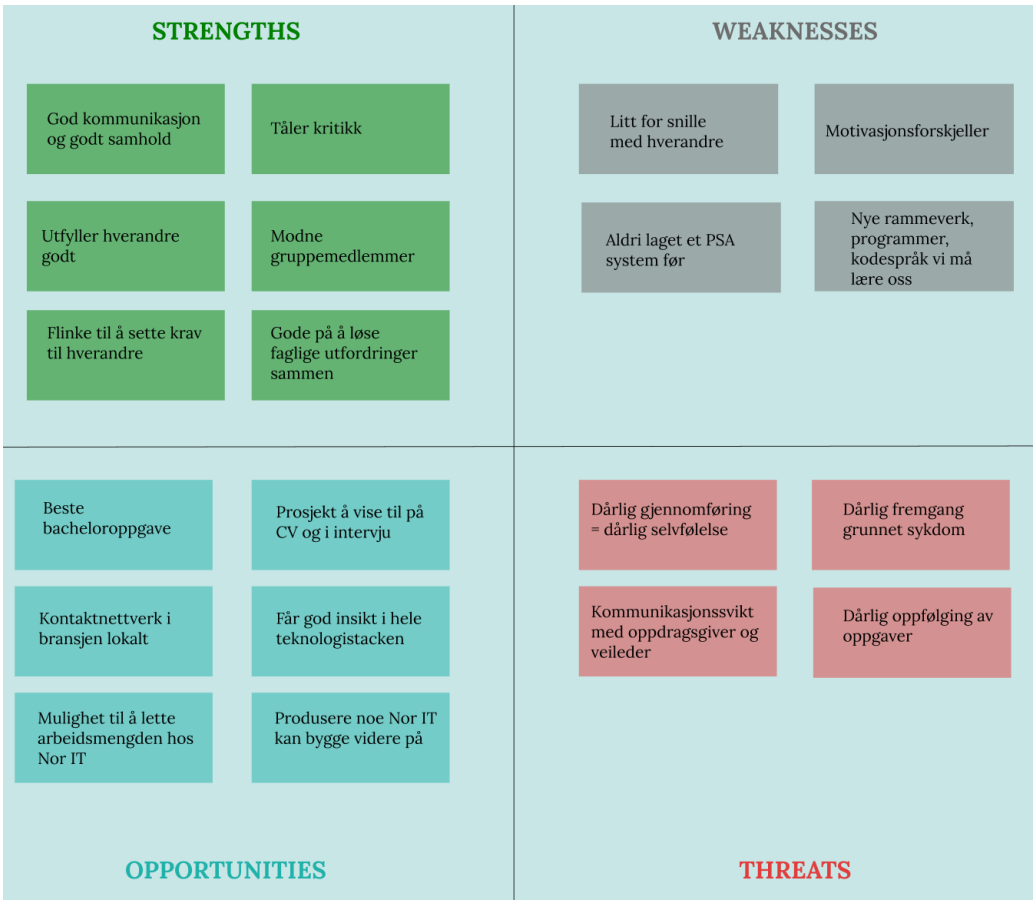
5.4.4 Andre

Docker er en teknologi for å legge programvare i isolerte miljøer eller kontainere som det kalles, med den hensikt å skape forutsigbare miljøer for programvaren å kjøre i og for å kunne dele/flytte programvare mellom ulike testmiljøer og servere. Vi benytter oss av docker fordi det gjør det lettere for oss å flytte programvaren vår mellom tjenester på Azure samtidig som det gjør det enklere for oss å bygge programvarens kjøremiljø ved å benytte oss av ferdigbyggede miljøer.

5.5 SWOT

Forkortelsen SWOT står for Strengths, Weaknesses, Opportunities og Threats (Vikøren & Pihl, 2022). En SWOT er et verktøy som vil være fin og benytte seg av for å få en bedre oversikt i en spesifikk situasjon. Denne typen analyse er også kjent som en situasjonsanalyse,

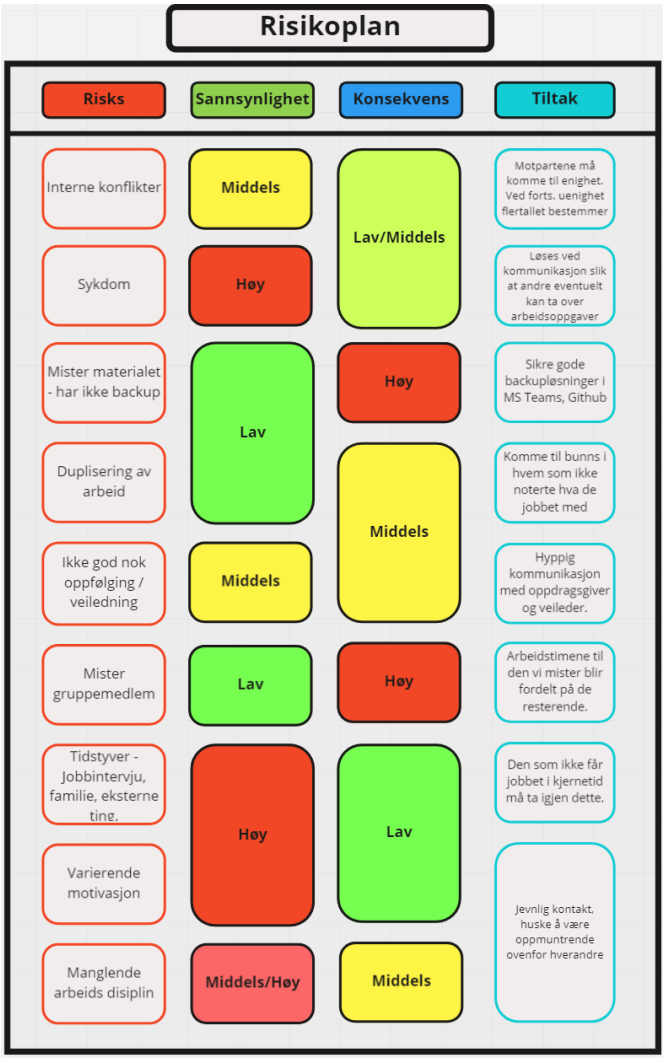
og vil hjelpe oss i dette bachelor prosjektet ved en oversikt over styrker, svakheter, muligheter og trusler som vist i figuren under.



Figur 6: SWOT-analyse

5.6 Risikoplan

Risikovurderingen vil være en systematisk gjennomgang av hva som for eksempel kan gå galt under et prosjekt. Denne vil brukes underveis i prosessen for blant annet å se om de tiltakene vi har satt opp vil være gode nok, eller om de må revideres underveis i prosessen (i.ntnu.no, u.å.). På figuren under vises risiko diagrammet vårt som består av det vi mener er situasjoner som kan oppstå under arbeidet vårt med prosjektet. Vi har valgt å bruke nivåene lav, middels og høy, med tanke på hvor sannsynlig det er at risikoen kan forekomme. Vi har også valgt å legge til en egen «tiltaks rad», hvor vi kort beskriver hva som kan gjøres for å unngå at risikoen forekommer, eller minsker graden av risikoen.



Figur 6: Risikoplan

5.7 Estimert tidsforbruk for prosjektet

Med tanke på estimatene vi har gjort, har vi resonert oss frem til forventet timebruk med en tanke på hvilke erfaringer vi har gjort oss ved tidligere prosjekter. Hvor vi har sporet tiden vi har brukt på de ulike prosjektene. Tidene er estimert gjennom plenumssamtale hvor vi systematisk har gjennomgått hver enkelt fase hver for seg og sett på de ulike oppgavene som hører til i fasen. Disse oppgavene har blitt definert som oppgaver på vår Kanban tavle i ClickUp. Det å kunne estimere tidsbruk på en god måte er noe som vil komme med erfaring. Siden vi sammen har kommet frem til kan du se i tabellen under har vi prøvd å gjøre det enklere og mer målbart for oss selv ved å dele tiden opp i fem ulike faser, vi ser i retrospektiv at kanskje det kunne vært fordelaktig og delt den opp enda litt mer, men vi hadde allerede kommet såpass langt i prosjektet og mener selv vi har ganske god kontroll på gjenstående tid og hva det vil kreve av oss.

Aktivitet	Startdato	Varighet	Estimert avsluttet	Faktisk avsluttet	Timeantall
Planlegging	28.12.2021	4 uker	21.01.22	26.01.22	100
Analyse	12.01.2022	3 uker	02.02.22	02.02.22	110
Design	28.01.2022	3 uker	18.02.22	16.03.22	550
Rapport	10.01.2022	18 uker	16.05.22	19.05.22	900
Implementasjon	21.02.2022	12 uker	16.05.22	-	800

Tabell 2: Estimert tidsforbruk

Det estimerte timeantallet fra USN er på 375t per gruppemedlem, som dere kan se i tabellen over har vi estimert timeforbruk for gruppen i sin helhet. Vi har estimert prosjektet til å ha en arbeidsmengde på ca. 410 arbeidstimer per person og 2460 timer totalt.

Design perioden gikk langt over hva vi hadde forventet og vi ser at dette er en del av prosjektet vi bør sette av mer tid til i fremtidige prosjekter. Grunnen til dette var at det var mye mer tidkrevende enn hva vi først så for oss, og få på plass alle elementene vi ønsket og få med. Dermed tok wireframe produksjonen svært mye tid, i tillegg til de andre delene av designfasen, slik som databas. På den andre siden har vi vært lure og jobbet med rapporten siden dag en, og dermed har vi hentet inn en del tid på den. Noe vi anser som veldig praktisk med tanke på at da har vi muligheten til å gjøre refleksjoner underveis i rapporten etter hvert som prosjektet tar form. Slik vil vi ikke miste eller glemme noe av informasjonen, siden vi jobber kontinuerlig med rapporten.

6 Analyse

6.1 MoSCoW

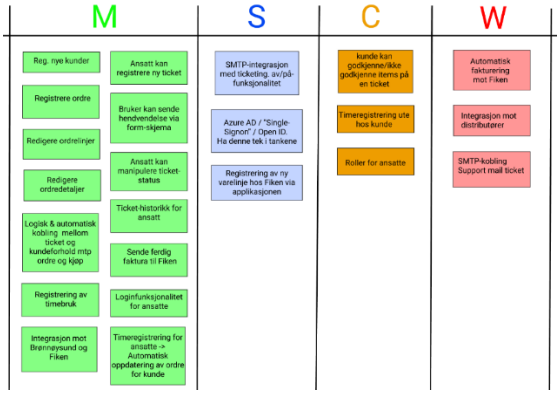
MoSCoW er et akronym som hjelper både oss og oppdragsgiveren med å se hvilke funksjoner som skal prioriteres, men også hva som er «fint og ha med» i tillegg til den delen som vi ikke skal ha med under dette prosjektet hvor vi utvikler systemet til Nor IT. Under kommer en kort forklaring hva de ulike bokstavene står for:

M – Must: Dette er en nødvending egenskaper som må være en del av systemet for at det ferdige produktet skal bli optimalt
S – Should: Dette er egenskaper som bør være med, om det er mulig og få til. Hvis man ikke har muligheten i innværende fase, bør man gå tilbake til de i fase 2.
C – Could: Dette er egenskaper som er ønskelige, men de kan fint kan bli utelatt fra nåværende fase. Har man ikke tid eller mulighet i denne fasen, kan man velge og ta opp

tråden på disse egenskapene i fase 2. Men de anses ikke like viktige som egenskapene som ligger under Should.

W – Wont: Dette er egenskaper som er helt valgfrie og som er avtalt med oppdragsgiver at ikke skal være med i denne fasen. Det kan til og med være at de aldri vil bli inkludert i systemet noen gang, men er tatt med for å kunne vurderes i fremtiden og andre faser av prosjektet (Stephens, 2015, s.57).

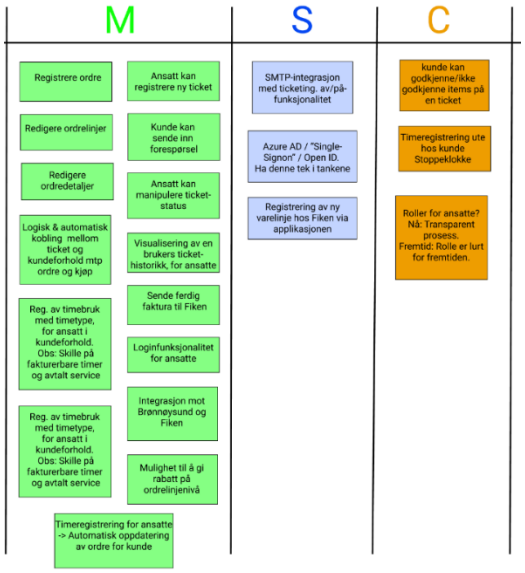
I denne MoSCoW-en er det også elementer som har vært ansett som nødvendige tidlig i prosjektet, men som har blitt endret, byttet ut eller fjernet underveis. Vi kan nevne elementer som innlogging for kunde har blitt tatt vekk for å kunne gjøre kommunikasjonen med kundene mer flytende, slik at kunden ikke har noe behov for å logge inn på hjemmesiden til Nor IT for å kunne komme i kontakt med sin kundebehandler. I stedet har vi gått over til å implementere epost i ticketen, slik at det vil være enklere for kunden når det kommer til kommunikasjonen med de ansatte. Hvorpå de ansatte heller ikke vil ha et behov for å gjøre notater fra epostene inn i ticketen, men all korrespondanse med kundene vil ligge på et sted. Den MoSCoWen du kan se under her er den overordnede, mens de to som ligger under funksjonelle / ikke funksjonelle krav har det blitt drillet ned på, slik at det som er overflødig har blitt fjernet.



Figur 7: MoSCoW - Overordnede

6.2 Funksjonelle krav

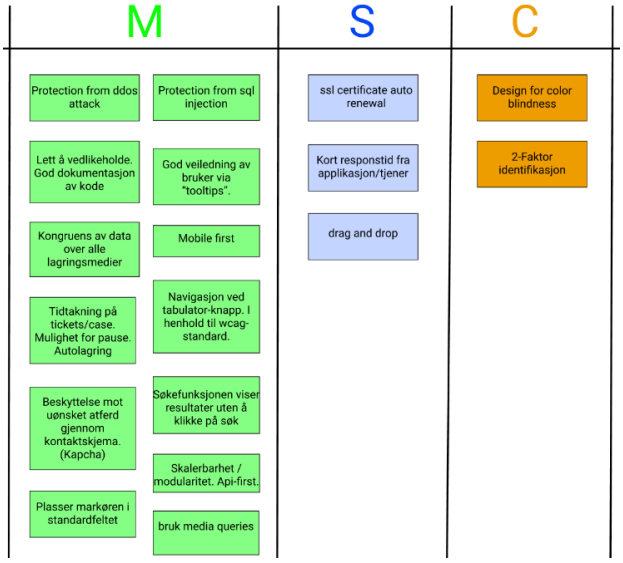
I MoSCoWen under har vi drillet ned for å luke bort de punktene som det ikke var behov for. Slik at vi kun står igjen med *funksjonelle krav* fordelt på M «må ha med», S «burde ha med» og C «kan ha med». Dette er en kravs oversikt som vil definere adferden og funksjonene som programmet vil inneholde (Stephens, 2015, s. 63).



Figur 8: MoSCoW - Funksjonelle krav

6.3 Ikke funksjonelle krav

Under kan du se hva vi står igjen med av *ikke funksjonelle* krav. Dette vil være kriterier gitt ved kvaliteten på systemet og koden, drift av programmet og bruken av det. Dette vil være krav som blant annet omfatter universell utforming, og sikkerheten (Stephens, 2015, s. 63).



Figur 9: MoSCoW - Ikke funksjonelle krav

6.4 Use cases

Use case eller på norsk kalt et bruksmønster er en beskrivelse av interaksjoner som foregår mellom ulike aktører. Disse aktørene kan for eksempel være enten brukere, ansatte, kunder eller deler av et system. Use cases lages for eksempel for å enklere se hvordan et spesifikt krav eller ønske bør implementeres, og ikke minst hvordan man skal håndtere avvik. En enkel mal for en Use Case bør inneholde tre hoved komponenter.

Tittel: Her vil være et beskrivende navn for hva use casen skal gå ut på, dette navnet inneholder helst både handlingen som skal utføres, og hvem som er hovedaktøren. Det andre trinnet vil være

Hovedscenario (Grunnleggende flyt av hendelser): Dette går ut på å lage en nummerert liste over ulike trinn som på best mulig måte beskriver den meste vellykkede måten og utføre den utvalgte casen på. Man får tastet inn korrekt bruker navn og passord, eller man søker på et korrekt fakturanummer for å finne den fakturaen som skal sendes til kunde. Det siste trinnet er

Utvidelsen av casen (Alternativ flyt av hendelser): Dette vil vise til hvis brukeren legger inn data som er ugyldige, for eksempel feil brukernavn eller passord, nettsider som er nede, eller hvis man søker etter en ordre med feil ordre nummer (Stephens, 2015, s. 78).

Etter hvert som prosjektet vårt har tatt form er det flere av use casene vi har produsert som ikke lenger er del av det systemet som blir slutt produktet vårt, disse er blitt lagt-til/arkivert under vedlegg. Vi så for oss at de var en del av systemet når vi var i analyse fasen vår, men etter møte med oppdragsgiver Olav på Nor IT den 16.02.2022 ble det avklart at kommunikasjon med kunden og dermed innlogging for kunde ikke nødvendigvis trenger og foregå igjennom ticketing systemet og dermed ikke vil være en del av fase 1, altså vårt oppdrag og utvikle. Det vil eventuelt bli en del av systemet som Nor IT selv ønsker å utvikle videre etter leveringen av vår bachelor prosjekt. Dette baserer seg på tidsfristen vi har og at vi har allerede en kompleks oppgave og løse. Hvorpå dette elementet kan bli lagt til en senere fase av utviklingen.

Under kan man lese et tabelleksempel av en use case for prosjektet:

Hvordan behandles Tickets av de ansatte hos Nor IT. a	
Kort beskrivelse av use case	Hvordan behandles tickets av de ansatte hos Nor IT.
Aktører	Kunder hos Nor IT Ansatte hos Nor IT Web applikasjon (PSA)
Forhåndskrav	PSA web applikasjonen er operativ og fungerer som den skal. Ansatte hos Nor IT er på jobb. Ansatte hos Nor IT er kjent med ticketing systemet.
Grunnleggende flyt av hendelser	En eksisterende kunde kontakter Nor IT med forespørsel. En ansatt fra Nor IT oppretter en ny ticket på kunden. Ticket får status åpen ticket.

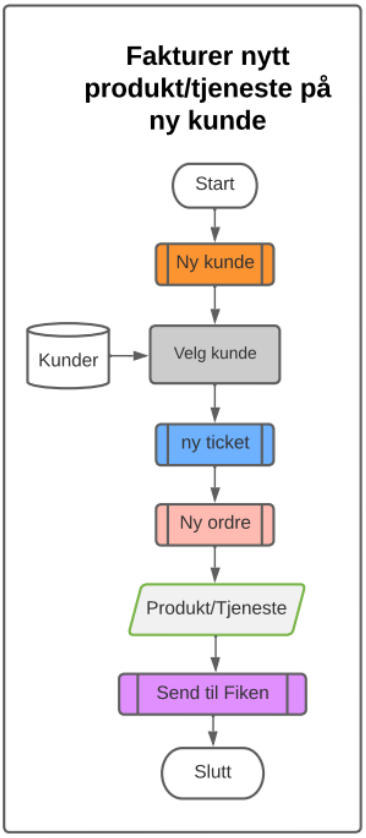
	En ansatt legger inn nødvendig informasjon i ticketen, videre får kunden svar via e-post. Kunden leser besvarelsen fra den ansatte ved Nor IT. Kunden føler hen fikk svar og takker for hjelpen. En ansatt lukker ticketen og markerer den som løst.
Alternativ flyt av hendelser	Nettside er nede eller undervedlikehold. De ansatte får ikke tilgang til PSA systemet. *Alternativflyt 2.0* En ansatt lukket en ticket ved uhell før den var ferdig.
Nøkkel scenarier	Nettsiden er nede eller drives vedlikehold på. Den ansatte glemmer å opprette ticketen og bruker Post-it lapp i stedet.

Tabell 3: Use Case - Ticketbehandling av ansatt

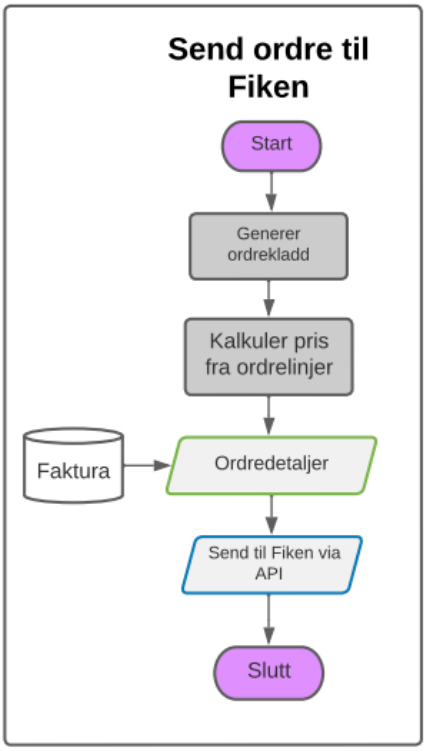
6.5 Dataflyt

Dataflytdiagrammer (DFD) gir oss en beskrivelse av systemet og hvordan prosessene samhandler med dataene (Stephens, 2015, s. 105). Dataflytdiagrammer hjelper med å vise oss hvordan dataene forflytter seg i systemet gjennom prosesser. Det som kan være vanskelig å forklare med ord kan ved hjelp av et godt utformet dataflytdiagram gi oss som utviklere en bedre forståelse på hvordan vi ser for oss hvordan systemet forflytter data fra punkt x til punkt y. Dette bidrar til å gi oss et visuelt fortrinn når vi på et senere tidspunkt skal kode dette. For mer avanserte dataflytdiagrammer som driller seg ned fra et kontekts-diagram er det lurt å bruke leveler på diagrammene som sier noe om hvor dypt i systemet man driller seg ned. Levelene består av: Level 0 (kontekts-diagram), level 1 (går ned på prosessene i kontekts-diagrammet) og level 2+(går et nivå enda lenger ned på prosessene som er i et level 1 diagram) (Bangerter, u.å.).

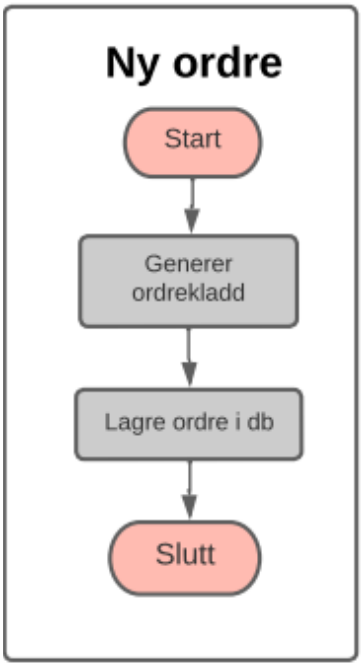
Underliggende ligger 3 dataflytdiagrammer for webapplikasjonen. De er henholdsvis level 1 og 2-diagrammer av prosessen fakturering hvor de to siste er level 2 som driller seg ned på «Fakturerer nytt produkt/tjeneste på ny kunde». Gjennom prosessen har det dukket opp elementer som har vært nødvendige for webapplikasjonens sin funksjonalitet og det har derfor vært nødvending med forandringer (se vedlegg for utgåtte diagrammer, samt diagrammer som ikke ligger i rapporten, men fortsatt er aktuelle)



Figur 11: DFD level 1 - Fakturere kunde



Figur 10: DFD level 2 - Sende ordre



Figur 13: DFD level 2 - Ny ordre

6.6 Hosting

Vi har vurdert flere mulige host løsninger for prosjektet. Deriblant Amazon Web Services (AWS), Google Cloud Platform (GCP), Azure og samt lokalt hos Nor IT på on-premis servere. Vi fant tidlig ut at å vi ville se til skyen og en av de løsningene slik at on-premis ble utelukket tidlig. Dette var også et ønske fra oppdragsgiver. Ettersom Nor IT er leverandør av Azure-løsninger og har god kompetanse på det var det naturlig at vi så ditt når vi begynte å teste løsninger. Etter å ha valgt leverandør av hosting måtte vi navigere valgene av type tjenester f.eks. virtuelle servere og maskinvare (VM) hvor man selv er ansvarlig for ressursbruk og allokasjon eller isolerte tjenester som Azure Wep APP hvor Azure i stor grad styrer hva og hvordan man har tilgang på server ressurser. I skyen er dette et av de viktigste punktene å være nøye i vurderingene på. Til en VM er det store kostnader forbunnet og det er vanskeligere å skalere uten å måtte sette opp utviklermiljøet helt fra bunnen av. Azure Web APP har mindre brukerfrihet, men også lavere kostnad og forenklede skaleringsprosesser. De er også tilrettelagt for kontainerløsninger slik som docker. Oppdragsgiver hos Nor IT veiledet oss også til å tenke mot mikrotjenester og det er noe som er enklere å sette i stand med

isolerte tjenester, heller enn å styre porter osv. på en VM. Azure Web APP lar oss også utvikle prosjektet med en cloud native struktur.

7 Design

7.1 Universell utforming av IKT

Når vi snakker om universell utforming betyr det at brukerne av nettsted, mobilapplikasjon, sosiale medier og mobilutgaver av nettsider, uavhengig av sine forutsetninger skal kunne ta i bruk de overnevnte nettløsningene på en god og enkel måte.

Det bidrar til å gjøre hverdagen enklere slik som for eksempel at videoer har tekst, slik at de kan benyttes på en god måte selv om man skulle være hørselshemmet, eller man oppholder seg i et støyende miljø, og ikke har tilgang på øreppropper eller headset til enheten (Digdir.no, u.å.). Skjermlesere er også noe man kan benytte seg av om man skulle være synshemmet, blind, eller ha andre utfordringer slik som dysleksi. En skjermleser er et program som tolker det som står i et dokument eller på en nettside eller applikasjon for så å formidle dette til brukeren via syntetisk tale også kalt (talesyntese). Skjermleseren vil dermed la synshemmede mennesker få tilgang til den digitale verden på en bedre måte, og bidra til å gjøre dem mindre avhengig av hjelp fra andre (Sandnes, 2018, s. 124). Det at teksten på nettløsningene er enkel å lese, og enkel å forstørre er også bra for synshemmede, og for de som har rotet bort lesebrillene sine. Det og kunne betjene nettsiden med tastaturet og ikke måtte være avhengig av mus, vil være bra for mennesker med permanent nedsatt motorikk, eller om man har vært uheldig og brukket armen! Med andre ord vil denne tilretteleggingen være helt nødvendig for noen i samfunnet, men det vil være bra for alle (Digdir.no, u.å.).

7.1.1 WCAG 2.0

WCAG (Web Content Accessibility Guidelines) er retningslinjer for hvordan nettsider skal lages for å gjøre de mer tilgjengelige. Deler av denne standarden er ikke bare anbefalt, men den er også lovpålagt både for privat og offentlig sektor. Bakgrunnen for å benytte seg av WCAG 2.0 når man utformer nettsider, er at innholdet skal bli tilgjengelig for flere brukere. WCAG 2.0 inneholder visse retningslinjer som skal gjøre sidene tilgjengelige for mennesker i samfunnet som kan ha ulike utfordringer. Slik som nedsatt eller manglende hørsel, nedsatt syn eller motoriske eller kognitive evner. De sammen retningslinjene vil også gjøre innholdet på de ulike nettsidene bedre for den generelle befolkningen også (Digdir.no, u.å.).

7.1.2 WCAG 2.0-Standaren

Forskriften om universell utforming av ikt-løsninger stiller krav om at både nettsider og applikasjoner skal og må oppfylle et minimum av 35 av totalt 61 kriterium i standarden. Disse deles inn i ulike nivåer, hvor nivå **A** er et «må-krav» mens nivå **AA** er et «bør-krav». Et lite utdrag av denne standarden kan leses i tabellen under (uutilsynet.no u.å.). Skulle man ønske å se kravlisten i sin helhet ligger denne lett tilgjengelig på uutilsynet.no.

Tittel	Kort beskrivelse	Nivå
Ikke-tekstlig innhold	Her skal man gi brukeren et tekstalternativ for innhold som ikke er tekst	A
Kontrast	Kontrastforholdet mellom bakgrunnen og teksten skal minst være på 4,5:1	AA
Endring av tekststørrelse	Teksten skal kunne bli endret til 200% uten at brukeren minster innhold eller funksjonen blir nedsatt	AA
Tastaturnavigasjon	All funksjonalitet på nettstedet skal kunne benyttes ved hjelp av kun tastaturet	A
Overskrifter og ledetekster	Her skal man sørge for at ledetekster og overskrifter er beskrivende	AA

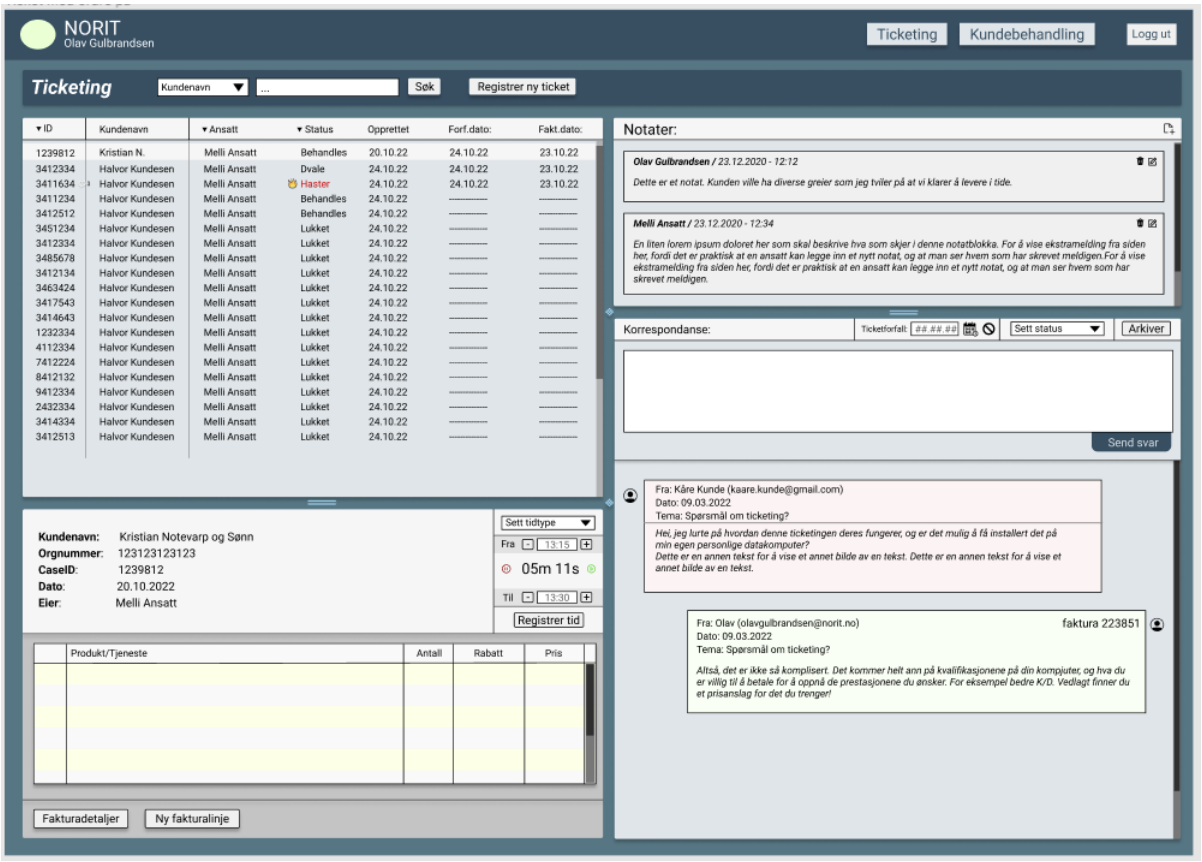
Tabell 4: WCAG 2.0 - Kravliste

7.2 Wireframes

Wireframes lager vi for å få en visuell forståelse og en god oversikt over de ulike sidene som skal inngå i systemet. Det hjelper oss og vise hvor på sidene de ulike komponentene skal ligge, og hvordan de ulike sidene skal se ut. Dette gjøres også for å få en universelt utrykk, slik at de vil være gjenkjennbare for brukeren. For dette prosjektet benyttet vi oss av Figma for å kunne lage detaljerte wirefames, siden det er mange komponenter som må på plass for å få systemet slik som oppdragsgiver og vi ønsker at det skal bli. Figma er også meget brukervennlig med tanke på samarbeid og vi hadde muligheten til å kunne jobbe med ulike design i samtid, og i samme «dokument».

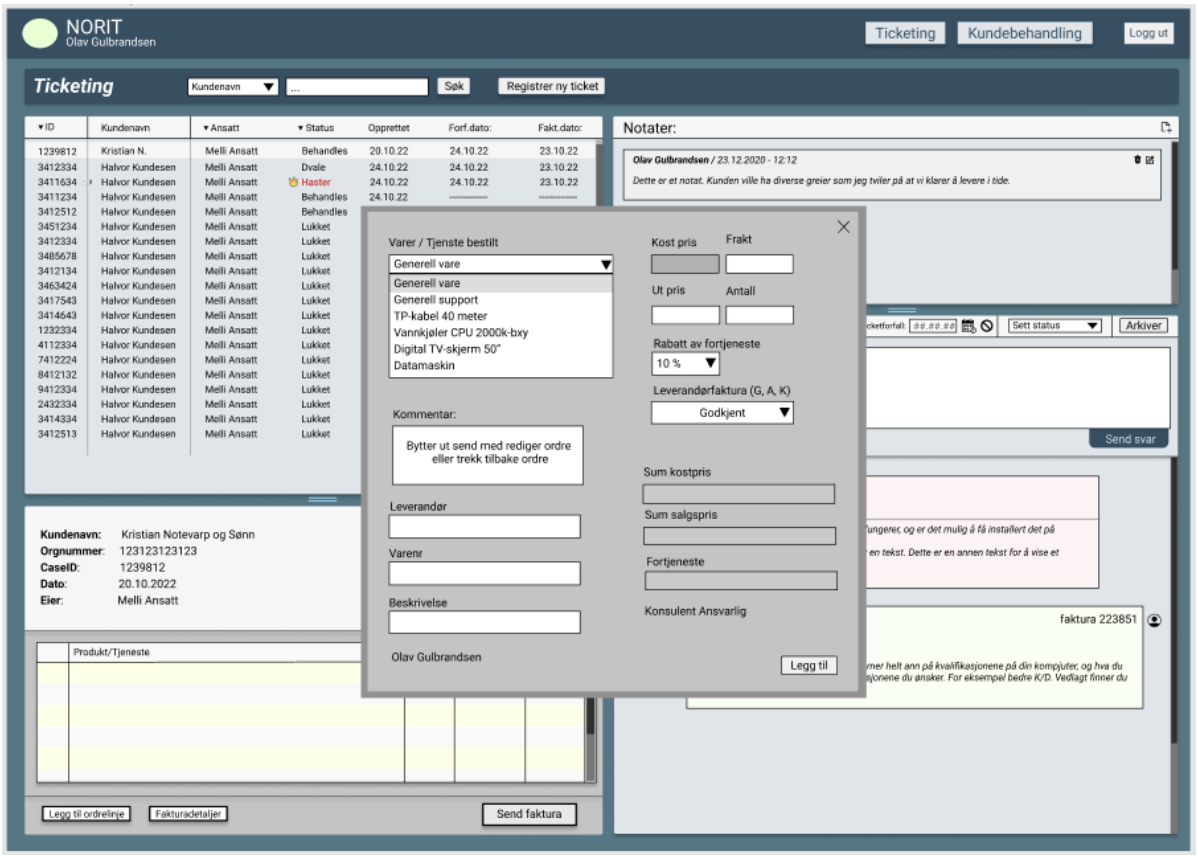
Vi har lagt ned mye tid i wireframsene under dette prosjektet, det har tatt mye tid og er noe av årsaken til at vi gikk over tiden vi hadde avsatt på designfasen. Det var mange elementer som skulle på plass, og vi gikk en del frem og tilbake både på utseende, men også når de kommer til funksjonaliteten for kontakten med kundene skulle foregå. I starten av prosjektet så vi for oss at kundene skulle logge inn i en egen portal for å kunne opprette, lese og svare på ticket. Vi har vært innom at kundene kunne få tilsendt en link på epost, for så å følge den linken for å kunne komme inn på deres ticket for å lese informasjonen som står der, eventuelt kunne besvare ticketen. Frem til slik systemet nå blir, hvor det kun er de ansatte som kan se ticketen, endre og gjøre notater i den, men epost-kontoene til Nor IT blir innlemmet i systemet, slik at

kundene kan bruke eposten sin som de vanligvis ville gjort, og har dermed ikke har noe behov for innlogging på eksterne sider. I tillegg til disse valgene, har vi også gjort vårt beste for å oppfylle ønskene oppdragsgiver har kommet med, for oss som aldri har jobbet med noe ticketingsystem før, så har det vært mye frem og tilbake på designbiten, men vi synes vi har landet et godt utgangspunkt for hvordan systemet kommer til å se ut til slutt. Hvis du skulle ønske å se på noen av de tidligere utkastene våres av wireframes, så ligger de under vedlegg. Den skjermdumpen du kan se under er av det siste utkastet av wireframen. På venstre side er det alle tickets i systemet, mens du kan se den øverste linjen er markert. Dette er den ticketen som står åpen i programmet. Under casene ser man, kundenavn, orgnummer, osv. Til høyre for dette er tidtakeren, dette vil se noe annerledes ut når systemet blir ferdig, med bakgrunn i at det ble for knotete og skulle endre på tiden direkte, dette vil nå være mulig og gjøre etter at tiden er stoppet, eller pauset. I boksen nederst til venstre vil ordrelinjer, antall, rabbatt og pris ligge. Mens hvis vi forflytter oss til øverst til høyre side vil det være interne notater, og under der igjen vil mail korrespondansen ligge.



Figur 12: Wireframe - Hovedside Nor IT webapplikasjon

Skjermdumpen under her, vil være hvordan de ansatte kan legge inn nye ordrelinjer, med kommentarer, priser, osv. Det vil også være muligheter for å legge inn rabatter.



Figur 13: Wireframe - Legge inn ordrelinje på ticket

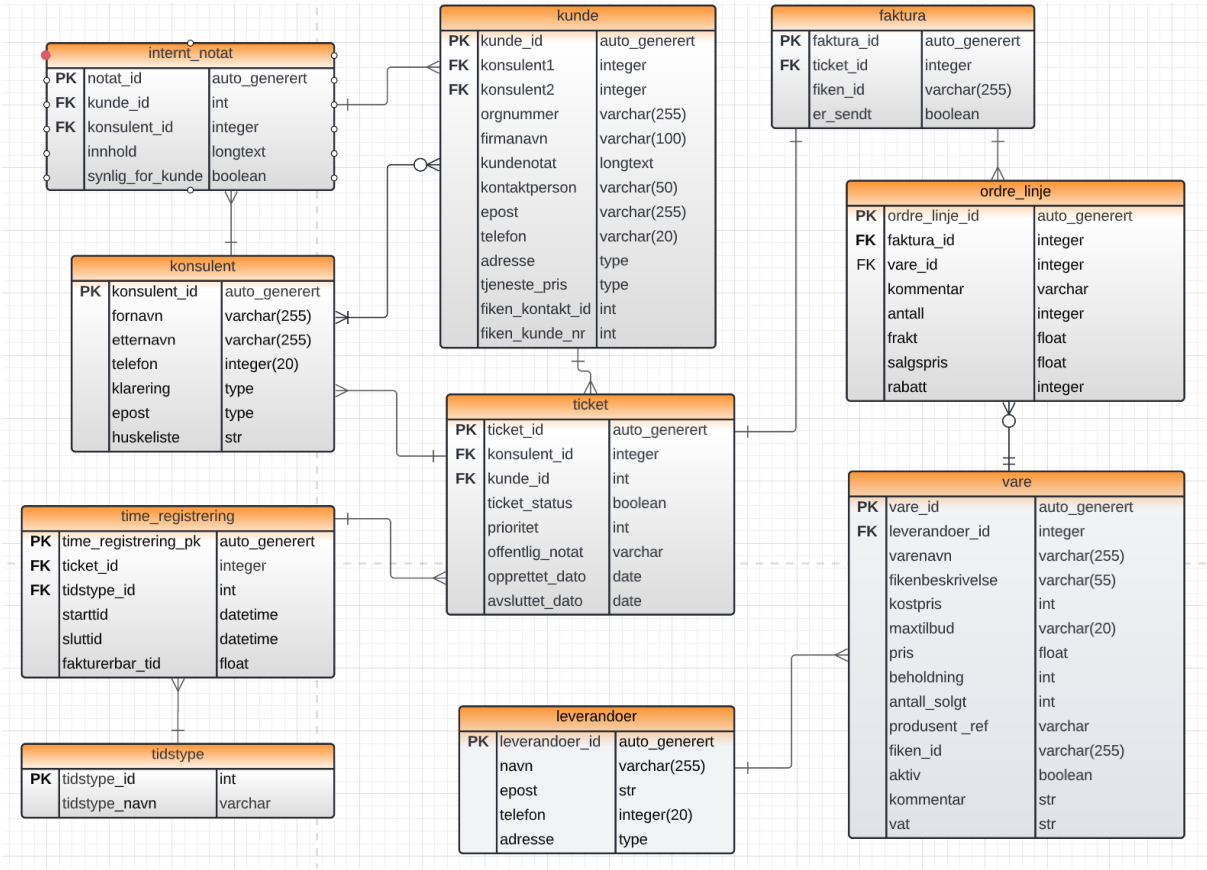
General data protection regulation (GDPR)

Den 20. Juli 2018 trådte personopplysningsloven i kraft i Norge, og den gamle loven fra 2000 ble opphevet. Forordningen sitt formål er i all hovedsak å sikre enkeltmennesker grunnleggende rettigheter og friheter, og da spesielt rett til vern rundt personopplysninger. Dette innebærer at disse personene man behandler ulike personopplysninger om, skal ha krav på å få vite akkurat hvilke opplysninger som blir lagret om seg selv. Dette er en plikt alle som håndterer informasjon som er personlig har. Informasjonen som gis angående dette skal være lettfattet og forståelig for alle. Det skal også være informasjon om hvor lenge personopplysningene skal lagres, hvor de skal lagres og hvem som har tilgang til dem. Personene kan også kreve at informasjonen om seg selv skal bli slettet (Gisle, 2018). Vi skriver om de valgene vi gjorde ang GDP i dette avsnittet.

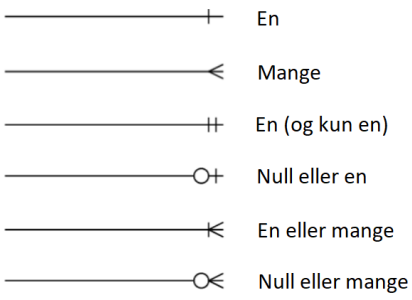
8 Databasemodell

I kapitlet Databasemodell, har vi lagt ved et bilde av vår modell, samt en kort forklarende tekst angående modellen og de valgene som har blitt gjort.

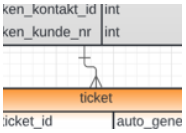
I vårt prosjekt benytter vi oss av en Relational Database Management System (RDBMS) eller på norsk relasjonsdatabase gjennom bruk av PostgreSQL. En relasjonsdatabase lagrer data i tabeller med relasjoner til andre tabeller. Script har blitt opprettet på bakgrunn av modellen med alle entiteter, tilhørende attributter, hovednøkler (PK) og fremmednøkler (FK) som vist i underliggende visuelle presentasjon av databasemodellen. Databasemodellen er et godt verktøy som gir oss et bilde av hvilke informasjon som vi ser på som essensielle og som skal være en del av databasen i denne fasen. Det å ha en slik visuell modell er for enklere å kunne se at designet er gjennomtenkt og at det er mindre sjanse for at det vil oppstå feil samt at det vil forenkle prosessen betraktelig hvis forandringer skal gjøres i fremtiden. Gjennom prosessen ved utviklingen av databasemodellen så har entiteter og attributter sett flere forandringer ettersom nye behov har dukket opp (tidligere versjoner kan sees i vedlegg). Vi har vært påpasselige ved å passe på at informasjonen som vi legger i databasen skal være relevant og at det ikke blir lagret mer data enn hva som er nødvendig som kan fort gjøre det uoversiktlig og skape redundans.



Figur 14: Databasemodell



Figur 16: Kardinalitet/Ordinalitet



Figur 155: Kardinalitet/Ordinalitet eksempel

Kardinaliteten og ordinaliteten er strekkene mellom entitetene eller tabellene som for eksempel «kunde» og «ticket». De forteller om hvordan forholdet mellom tabellene er, og hvor mange ganger en tabell kan refereres til i en annen tabell. I eksemplet med kunde og ticket så ser vi ved hjelp av figur 15 og 16 at en kunde kan ha «mange» tickets. Det vil si at en kunde kan ha flere instanser eller referanser hentet fra ticket. Men at én ticket kan kun være registrert eller referert på «en» kunde (Lucidchart, u.å.). Relasjonene i en RDBMS er viktige for hvordan logikken som skal ligge til grunne blir. Relasjonene i en database kan sees på tabellene hvor det er en FK som hentes eller refereres fra en PK i en annen tabell. Relasjonene i en relasjonsdatabase er viktig i den forstand at tilhørende data må ha en kobling. Hvis vi for eksempel skal legge inn en ny ticket på en kunde så må kunde kunne refereres på ticketen. Hvis det ikke hadde vært noe relasjon eller referanse ville det å legge inn en ticket uten noe form for kobling mot kunde gjort dataen redundant siden man vet ikke hvilken kunde denne ticketen er oppført på. Dette gjøres ved å ha FK «kunde_id» på tabellen ticket. Relasjoner bidrar også til å gjøre spørringer opp mot databasen og det å søke opp og hente data enklere. Et godt fundament som er oversiktlig vil også bidra til å forenkle prosessen ved å implementere forandringer når/hvis den tid kommer, selv for utenforstående personer med database erfaring.

På bakgrunn av valgene vi har vi spurt oss selv «hva er det Nor IT trenger». I det legger vi hvilken informasjon har de behov for i webapplikasjonen og dets funksjonalitet, og hva som skal til for at disse funksjonene har den informasjonen de trenger samt kunne legge inn, for eksempel ved opprettelse av en ny kunde.

Databasemodellen har vært gjennom flere revisjoner og det har forekommet en del forandringer underveis, i neste kapittel skal vi se på sikkerhet, et spennende og høyest aktuelt tema i dagens samfunn.

9 Sikkerhet

Dette kapittelet forteller om vår tilnærming til generell sikkerhet og autentisering i systemet, hvordan vi har sjekket etter sårbarheter og hvordan vi har sikret sensitiv data på GitHub.

Datatilsynet kan fortelle at informasjonssikkerhet omfattes av konfidensialitet, integritet og tilgjengelig, også kjent som CIA-triaden, samt robusthet. Hvor konfidensialitet innebærer at uvedkommende ikke får tilgang til informasjonen. Integritet skapes ved at man ikke ved utilsiktet bruk eller at uvedkommende får endret informasjon. Mens tilgjengelighet defineres med hvorvidt informasjonen er tilgjengelig ved behov og autorisasjon. Datatilsynet beskriver robusthet som systemet og organisasjonens evne til gjenoppretting (Datatilsynet, 2018).

Konfidensialitet

For å etterleve kravene til konfidensialitet har vi implementert autentisering på rotnivå for frontend og for tilgang til API-et. Mer om hvordan vi har gjort det kan leses om i delen om autentisering i dette kapittelet.

Integritet

Vi har gjort vårt beste for å sørge for å opprettholde integriteten i informasjonen ved å bruke f.eks. Object-relational mapping (ORM) via biblioteket SQLAlchemy som kommer innebygget med SQL-injection beskyttelse. Vi har også beskyttet API-et vår mot feil input ved å definere hvilke type data som kan sendes til den og håndtere det om dataen bryter med definisjonen.

Tilgjengelighet

For oppdragsgiver var det ikke aktuelt med noe gradering av tilganger i systemet på nåværende tidspunkt da de ønsket å arbeide transparent slik at alle ansatte hadde innsyn i alle aspekter av driften. Det ble påpekt at gradering kunne bli aktuelt i fremtiden ved videre vekst av selskapet og vi har derfor lagt til rette for gradering med attributt i databaseklassen for konsulenter hos oppdragsgiver.

Robusthet

I vårt system skaper vi robusthet ved å lene oss på mulighetene skyen har å tilby. Vi kjører alle våre webapplikasjoner som egne frittstående applikasjoner i Azure. Det betyr at ved kritiske feil hvor applikasjonen går ned vil den automatisk re-starte for å sikre maksimalt med oppetid. I tillegg kjøres de gjennom Docker som vil si at dersom noen får tilgang til

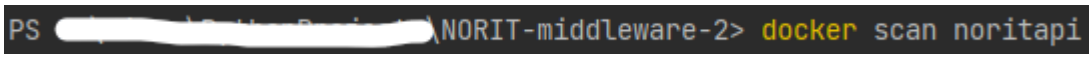
applikasjonen og får forandret kildekoden. Kan den enkelt slettes og settes opp på nytt innen rimelig tid. For API-et har vi også implementert logging. Det gir oss mulighet til å gjøre målrettede søk i et dokument for feilsøking dersom uhellet skulle være ute, hvilket også resulterer i hurtigere gjenopptagelse av driften i systemet.

Autentisering

Microsoft Azure Active Directory gir oss mulighet til å sette bort både en del av sikkerhetsaspektet, men også krav som faller innenfor GDPR rammen. Ved å la Microsoft ta seg av lagringen av passord og brukernavn sørger vi for at vi ikke er de ansvarlige for lagring av persondata av den typen. Den samme autentiseringen danner grunnlaget for hvordan vi autentiserer i API-et. Ved bruk av API-et sender man med en token som genereres av Active Directory, denne verifiseres ved en sjekk og om alt stemmer vil man da få tilgang til API-et. For API-et vil man da kunne forsikre seg om at forespørselen kommer fra riktig applikasjon. Om man da i tillegg whitelister personen i Active Directory for bruk av både frontend applikasjonen og API-et hver for seg må man gjennom flere steg gjøre dette for å kunne skaffe seg utilsiktet tilgang.

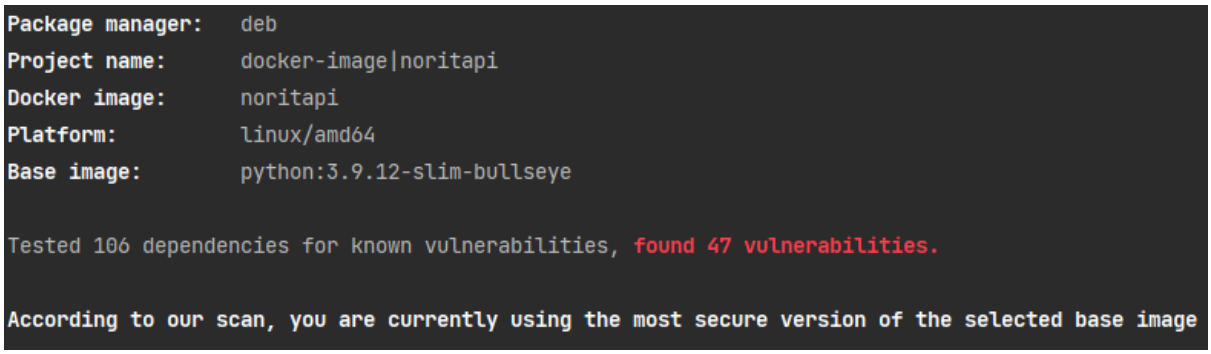
Sårbarhet

For å lete etter sårbarheter i applikasjonene våre benytter vi oss av Snyk. Snyk kommer innebygget i Docker og skanner etter kjente feil og mangler og kjøres på denne måten.



Figur 17: Sikkerhet - Docker snyk scan

For API-et vår vil det gi dette kjøreresultatet.



Figur 18: Sikkerhet - Docker snyk kjøreresultat

Nå viser testen at vi bruker den best sikrede versjonen bildet Docker konteineren kjører på, som for så vidt er bra. Med unntak av en av sårbarhetene er de alle av lav alvorlighetsgrad. Den ene som stikker seg ut er derimot en kritisk sårbarhet.

```
× Critical severity vulnerability found in openssl/libssl1.1
  Description: OS Command Injection
  Info: https://snyk.io/vuln/SNYK-DEBIAN11-OPENSSL-2807596
  Introduced through: openssl/libssl1.1@1.1.1n-0+deb11u1, ca-certificates@20210119, krb5/libgssapi-krb5-2@1.18.3-6+deb11u1
  From: openssl/libssl1.1@1.1.1n-0+deb11u1
  From: ca-certificates@20210119 > openssl@1.1.1n-0+deb11u1 > openssl/libssl1.1@1.1.1n-0+deb11u1
  From: krb5/libgssapi-krb5-2@1.18.3-6+deb11u1 > krb5/libkrb5-3@1.18.3-6+deb11u1 > openssl/libssl1.1@1.1.1n-0+deb11u1
  and 1 more...
```

Figur 19: Sikkerhet - Docker snyk risikofunn

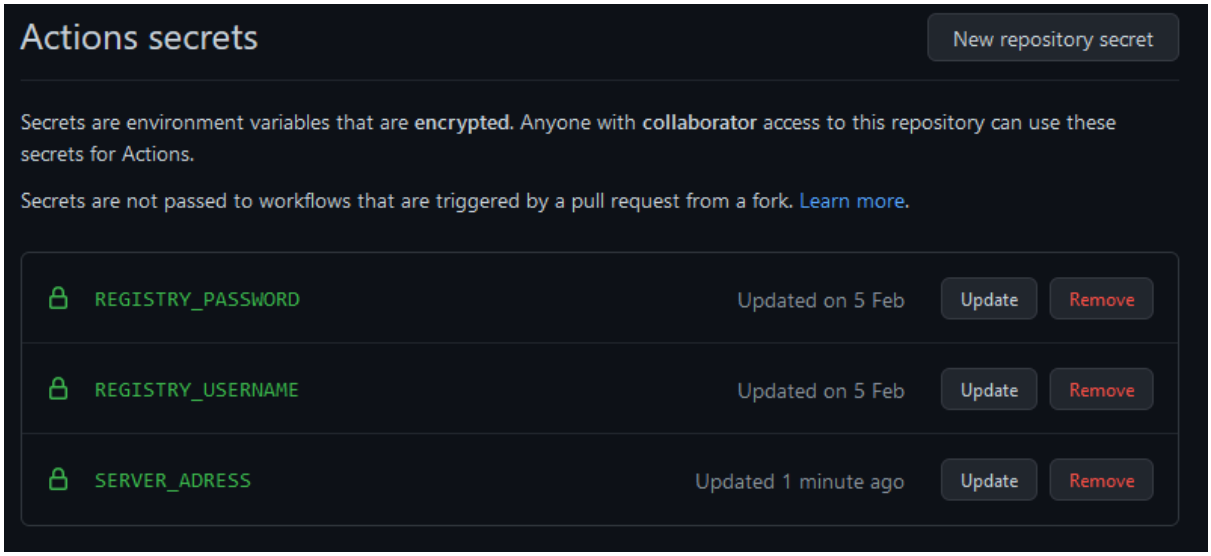
Dette er noe som må utbedres, og vi må på veien videre gjøre forandringer i Docker konteineren.

Sikring av kode på GitHub

Koden til systemkomponentene ligger i private repositories på GitHub. Det vil si at ingen i utgangspunktet skal kunne få tilgang til koden uten invitasjon. Dersom det skulle skje at noen fikk tilgang til den vil passord og tokens vi benytter oss av være skjult ved bruk av GitHub secrets som vist under ved eksempel fra koden som automatisk publiserer koden til Azure serveren. Secrets kan ikke avleses i GitHubs portal, slik at man trygt kan gi tilgang til koden uten at man eksponerer sensitive data, passord eller nøkler

```
29              login-server: ${{ secrets.SERVER_ADRESS  }}
30              username: ${{ secrets.REGISTRY_USERNAME  }}
31              password: ${{ secrets.REGISTRY_PASSWORD  }}
```

Figur 20: Sikkerhet - GitHub



Figur 21: Sikkerhet - GitHub actions secrets

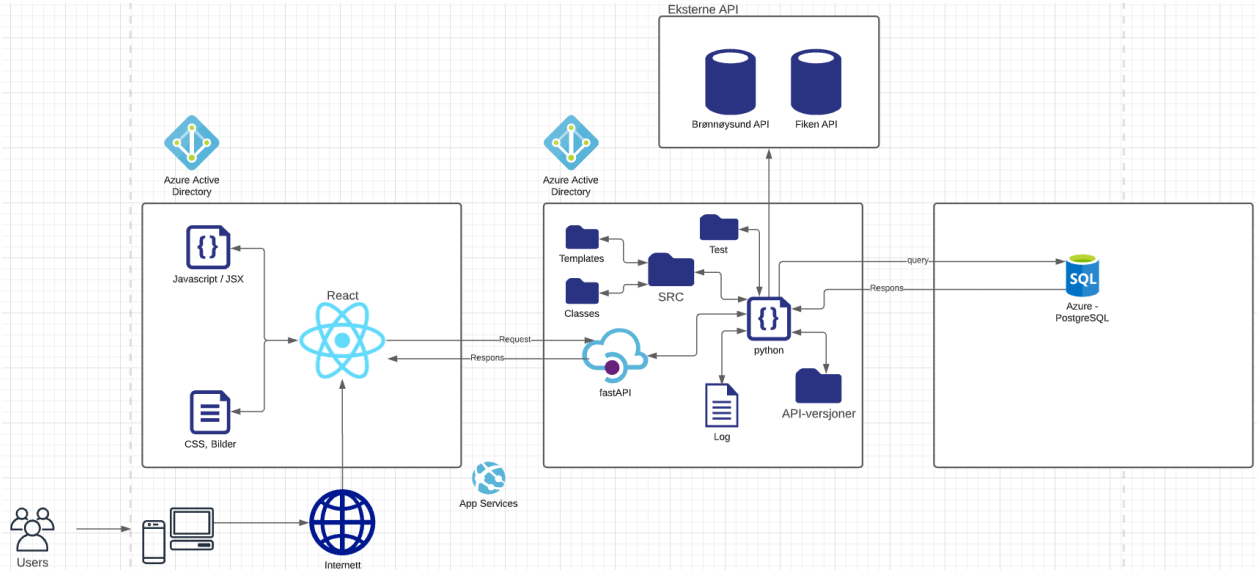
10 Systemarkitektur

I kapitlet om systemarkitektur kommer vi med en grafisk illustrasjon av vårt «roadmap», som gir oss en oversikt over prosjektet.

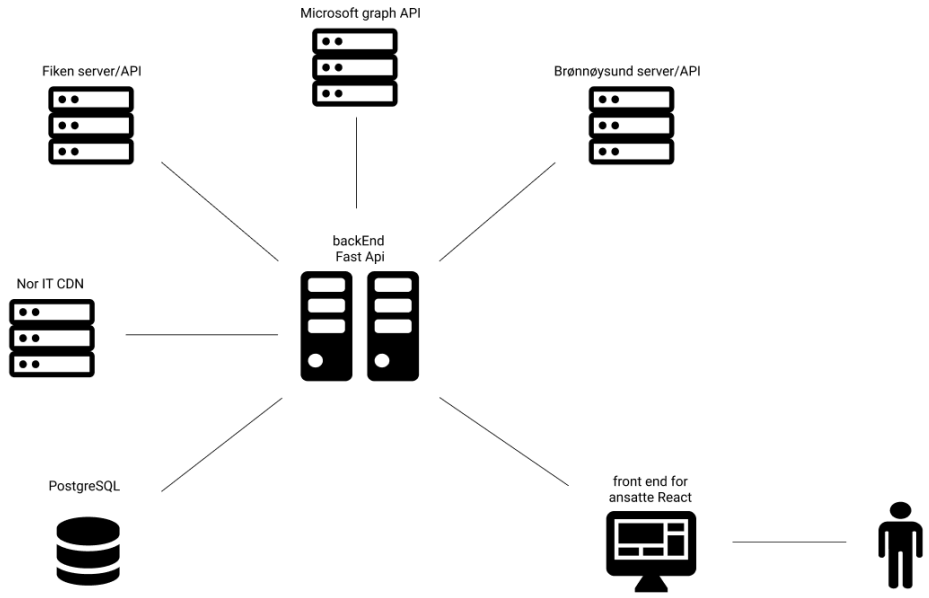
Systemarkitektur er en modell som skal fungere som et «roadmap» for applikasjonen vi lager. Kort forklart tenker vi å ta i bruk Azure Active Directory som skal håndtere identitet og tilgangskontroll i prosjektet vårt, React som hoved teknologi, PostgreSQL som database og fastAPI som API rammeverket. Bakgrunnen av valget vårt ved bruk av Azure Active Directory er med hensyn til sikkerhet i applikasjonen, Azure Active Directory er utviklet av Microsoft.

Når vi skal utvikle systemet får vi ved hjelp av system arkitekturen i figurene under en god oversikt over hvordan vi tenker systemet skal være delt inn (se vedlegg for utgått versjon). Figur 23 viser en mer detaljert visuell representasjon enn figur 24. Front-end blir utviklet ved hjelp av react hvor kode, filer og bilder blir inndelt i mapper som er kategorisert og organisert. Ved inndeling og med god semantikk blir lesbarheten samt vedlikehold vesentlig lettere. Ved inndeling vil det også forenkle prosessen med logging for kunne se historikk på mindre områder av systemet for å lettere kunne feilsøke hvis eventuelle feil dukker opp. I back-enden bruker vi en samme type mappedeling for de samme argumentene som er blitt nevnt. Ved å dele koden ved at vi lager klasser på de ulike komponentene vil det også være lettere for integrasjon på tvers av systemer ved å kun bruke de komponentene man har behov for. API-et vårt blir utviklet med rammeverket fastAPI som benytter Python kode som skal

fungerer som middlewaren vår som vil få forespørsler fra front-end og DB som igjen sender tilbake relevant respons fra forespørslene. Vi vil bruke to eksterne API-er som går til Brønnøysund og fiken. Disse API-ete vil vi ikke kunne påvirke på noe som helst måte, det er noe de styrer selv, vi kun tar de i bruk.



Figur 22: Systemarkitektur diagram (detaljert) for Nor IT webapplikasjon – tredje utkast



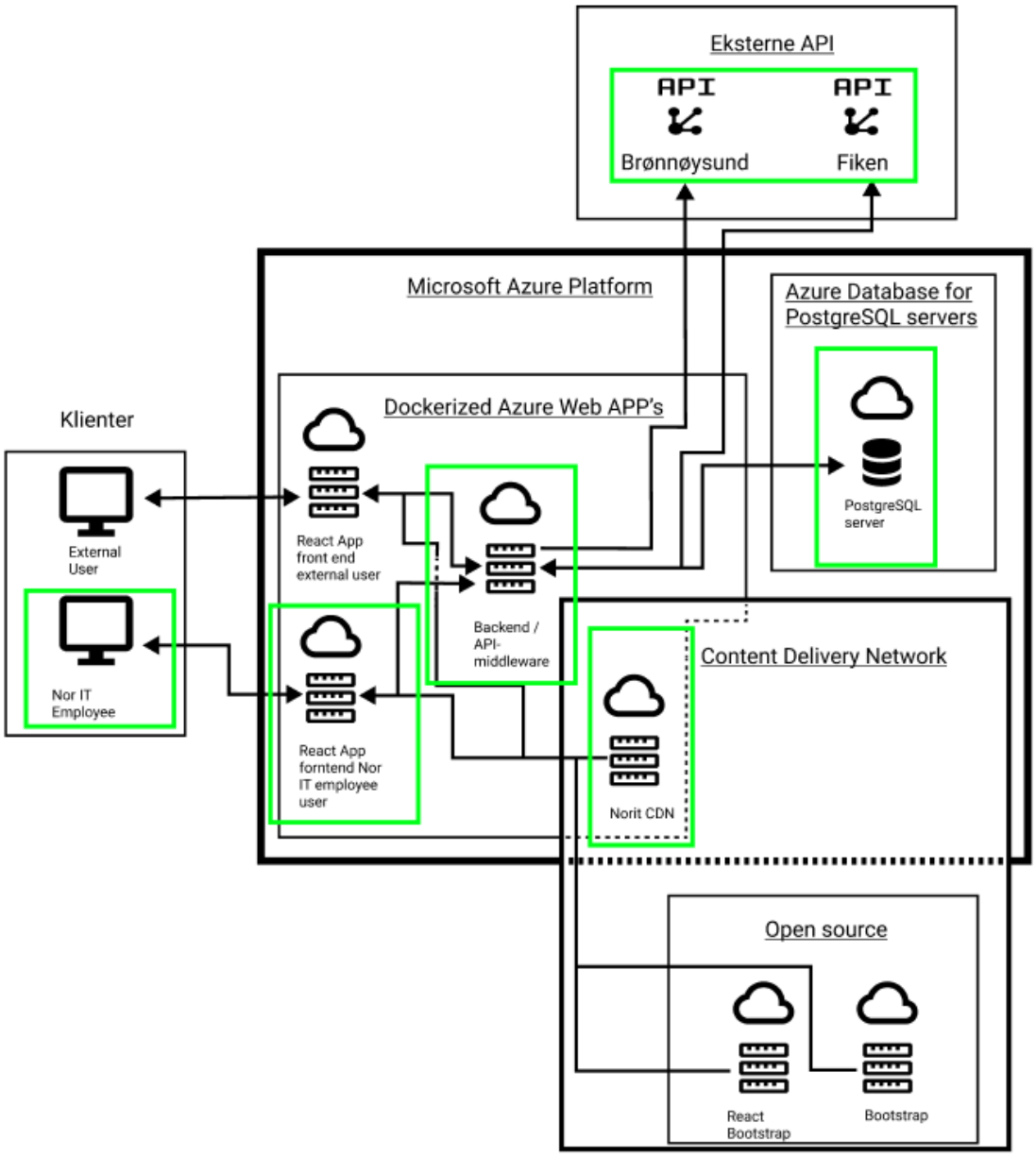
Figur 23: Systemarkitektur diagram for Nor IT webapplikasjon – andre utkast

11 System Infrastruktur

På likhet med systemarkitektur kapitlet skal vi her se på infrastrukturen til prosjektet, infrastrukturen skal gi ett overblikk over komponentene vi benytter oss av.

Infrastrukturen viser hvilke fysiske komponenter, så vel som applikasjonskomponenter systemet er bygget opp av og hvordan disse henger sammen. Ettersom vi har valgt å benytte oss av en skybasert løsning for hosting av systemet slipper vi å forholde oss til nettverkskomponenter. Det hadde vært annerledes om vi selv sto for hostingen. Vi har valgt å legge alle applikasjonskomponentene våre i docker kontainere og hoste disse i tjenesten Azure Web App. Dette har vi gjort fordi vi da får den samme bygge og publiseringsprosessen på uavhengig av teknologiene de er bygget på. Det koster litt mer tid i oppsett av CI/CD strukturen, men sparer oss mye hodebry i det lange løpet. Vi har også valgt å benytte oss av Content Delivery Networks (CDN) i form av open source biblioteker på frontend delen av systemet. Dette fordi det sparer oss for å måtte designe og kode all UI fra bunnen av. Utover dette har vi valgt å modellere for, og produsere et eksempel på et internt CDN. Dette med tanke på videre skalerbarhet og sikkerhet. I fremtiden kan man flytte sine komponenter inn i et internt CDN og forenkle dokumentasjonsprosessene, implementasjon på tvers av ulike plattformer samt at man fjerner mulige sikkerhetsbrister ved å bruk av ekstern kode. Databasen hostes også på Azure, men da på tjenesten Azure Database for PostgreSQL servers. Dette har sine fordeler og ulemper. Den største ulempen er at man ikke har tilgang på rotbrukeren på databasen mens de store fordelene er at all hosting er samlet på Azure slik at man enklere kan sette opp hvilke tjenester som har tilgang til den. Dette og flere andre sikkerhetsperspektiver er Microsoft mye sterkere enn oss på og det tjener oss best å la de styre det både med tanke på tidsbruk og faktisk sikkerhet.

Diagrammet under viser komponentene vi har sett for oss at systemet vil kreve på sikt, men det er kun elementene med grønn innramming vi fokuserer på å implementere.

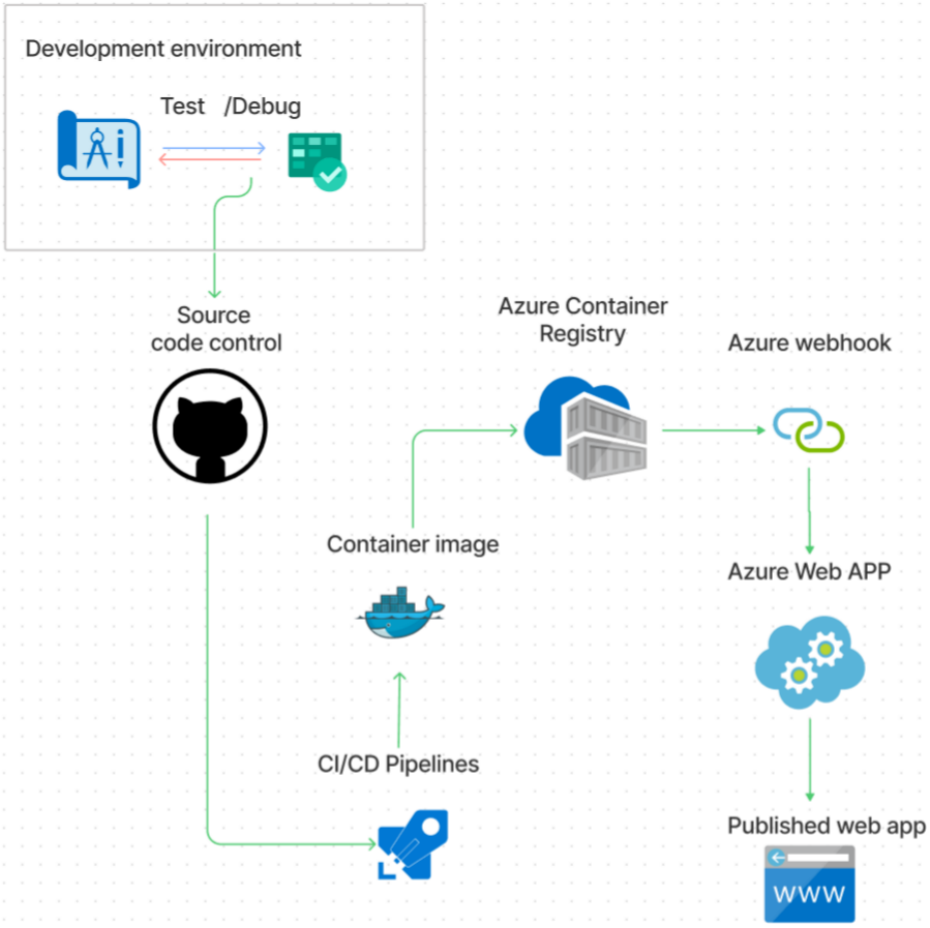


Figur 24: System infrastruktur

I kapittel 12 skal vi se nærmere på integrasjon og leveranse.

12 Kontinuerlig integrasjon og leveranse

Integrasjon og leveranse er en viktig del av oppgaven for oss og oppdragsgiveren, i dette kapitlet sier vi litt om det og prosessen rundt det.



Figur 25: CI/CD diagram

Det har vært viktig for oss å ha en smidigst mulig implementasjon av kode ettersom vi har to store komponenter i vårt system som må kunne snakke med hverandre. Vi har derfor valgt å bruke noen ressurser på å sette oss inn i devops metodikken CI/CD slik at vi på en god måte kan teste integrasjon av komponenter på tvers av systemet uten å måtte oppdatere lokale versjoner av koden, men heller teste mot nyeste kode plassert i skyen. Det tillater oss å kunne isolere arbeidsoppgaver og gruppere oss slik at noen arbeider med logikk i backenden og andre jobber med logikk eller brukergrensesnitt på frontenden. Det gjør det også enklere for oss å samarbeide om hvilke komponenter som arbeides med og som må ferdigstilles. Et eksempel vil være at de som arbeider på frontenden «bestiller» en route for å ta imot og behandle noe spesifikk data på backenden. Når denne så blir produsert vil backenden kunne

melde fra at funksjonen nå er live. CD biten av CI/CD lar oss kontinuerlig forsikre oss om at koden er kjørbar i ønsket miljø på Azure.

13 Testing

I dette kapittelet vil du få innblikk i metoder og verktøy vi har benyttet for testing av komponentene i systemet gjennom teori og skjermbilder fra gjennomførte tester.

13.1 Test metodikk og verktøy

For å teste systemet vi produserer på en måte som lar oss verifisere og kvalitetskontrollere brukergrensesnitt og bruksmønster, samt kodekvalitet opp mot oppdragsgivers forventninger og krav har vi valgt å dele våre testprosesser i fem deler. Disse er brukerreise, end-to-end, API-route tester, sikkerhetstester og kodetester.

Brukerreise

For å kvalitetssikre og teste brukerreiser i systemet vårt har vi primært kjørt jevnlige samtaler med oppdragsgiver for å sikre oss at det vi lager er det de ønsker. Gjennom denne prosessen har vi også produsert en interaktiv prototype i Figma hvor oppdragsgiver har hatt anledning til å komme med innspill. Denne prosessen lar oss avstemme UI-avgjørelser fortløpende i samarbeid med oppdragsgiver

End-To-End

End-to-end testing er tester som gjøres for å kontrollere at brukeropplevelsen og brukeren på siden følger en logisk struktur. Vi har da valgt oss ut use cases eller user stories som vi har fulgt fra start til slutt for å se at 1. all funksjonalitet er på plass 2. Elementene følger hverandre i logisk rekkefølge 3. Applikasjonsflyten er slik den er tiltenkt med tanke på tidsbruk og systemintegrasjoner osv.

API-Route Testing

For å forsikre oss om at våre endepunkter eller routes i API-et gir, sender og mottar riktige data har vi benyttet oss av verktøyet Postman. Vi har valgt å bruke et eksternt verktøy på tross av at API-rammeverket vi benytter oss av kommer med ferdigbygget dokumentasjon som også tilbyr tester av routes fordi vi da forsikrer oss om at systemet håndterer forespørsler fra utenforstående systemer på en ønsket måte.

Sikkerhet

For å teste sikkerheten i applikasjonen vår har vi benyttet oss av Snyk. Snyk er et rammeverk for sikkerhetstester og dette kommer innebygd i Docker. Det fungerer slik at det søker gjennom koden etter kjente sikkerhetsfeil og exploits. Vi benytter oss av et ferdigbygd rammeverk for dette er da noe vi ikke har kompetansen i teamet til å gjøre på en mer hensiktsmessig måte enn det Snyk kan gjøre for oss.

Kodetester

For å teste produksjonskoden vår har vi skrevet enhetstester og integrasjonstester. Enhetstester gjøres for å teste enkeltkomponenter slik som funksjoner og metoder eller klasser og klassemetoder. Integrasjonstester gjøres for å teste samhandlingen mellom flere slike enheter. Vi kunne også valgte å skrive spesifikke API tester, eller såkalt API-mocking for å teste intern funksjonalitet mot eksterne APIer. Dette har vi valgt å ikke gjøre fordi vi allerede har to manuelle metoder å teste dette på.

Developer Tool

Tidlig i produksjonsfasen lagde vi et enkelt verktøy som raskt kunne oppdatere vår database med data via vårt API. På denne måten fikk vi umiddelbar tilbakemelding på om vår applikasjon fungerte i samsvar med våre forventninger. Dette var til stor hjelp da det ga bedre arbeidsflyt gjennom arbeidsprosessen.

13.2 Test Design

For å holde orden på hva og hvordan vi skal gjøre UI/UX og brukerreise tester har vi laget et testdesign dokument hvor vi har ført inn tester som må gjøres, og evt. resultatet av disse.

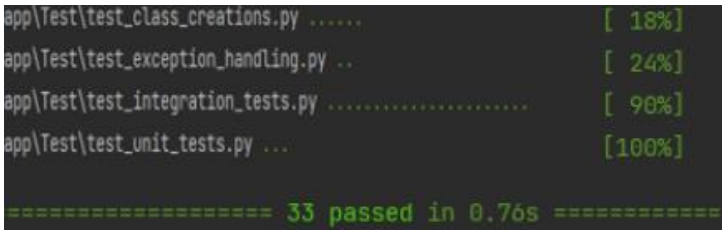
	Forhåndsbetingelser	Steg	Test Data	Forventet resultat	Oppnådd resultat	Ny/Alternativ test	story/case ref
2	Må ha installert Excel eller tilgang på programmet gjennom MS Teams	1. Gir testcasen en beskrivelse 2. Setter inn tekstboks for å beskrive stegene nødvendig for å gjennomfør testen 4. osv.	Kan være input data dersom man tester login, eller for JSON dict om man mocker api	Vellykket login - for login test statuscode 200 - for api Vellykket JSON parsing - for api osv.	Her beskrives det oppnådde resultatet med farger Om testen var oppnådde forventet resultat beskrives det med grønn tekst Om testen feilet beskrives det med rød tekst	Referanse til en annen test ID	referanse til en usecase eller userstory
3	Bruker må være logget inn	Åpner applikasjonen Velger notatfunksjon Noterer ned testdata Minimerer notatet Gjenåpner notat og Kontrollere at data ble lagret	Notatdata: "Hei, jeg er testdata klar til input i notat"	Applikasjonen åpnes Notat åpnes Notat tar imot data Notat lukkes Notat lagrer data Notat gjenåpnes Notat viser data	Alle forventede punkter fra forrige celle fungerer som ønsket	Ingen	#UC3

Dette dokumentet er laget i Excel og har ligget tilgjengelig på teamsområdet slik at alle har hatt mulighet til å løpende oppdatere det med nyttige tester.

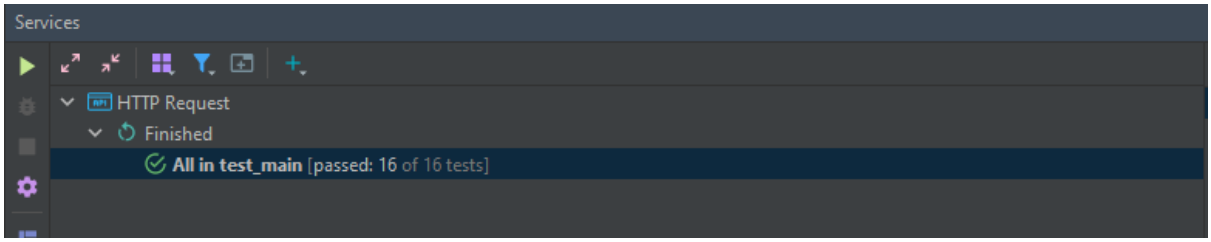
Figur 26: Test design - Eksempel fra test design dokumentet

13.3 Praktisk gjennomføring av tester

Eksempler fra kode og manuelle tester



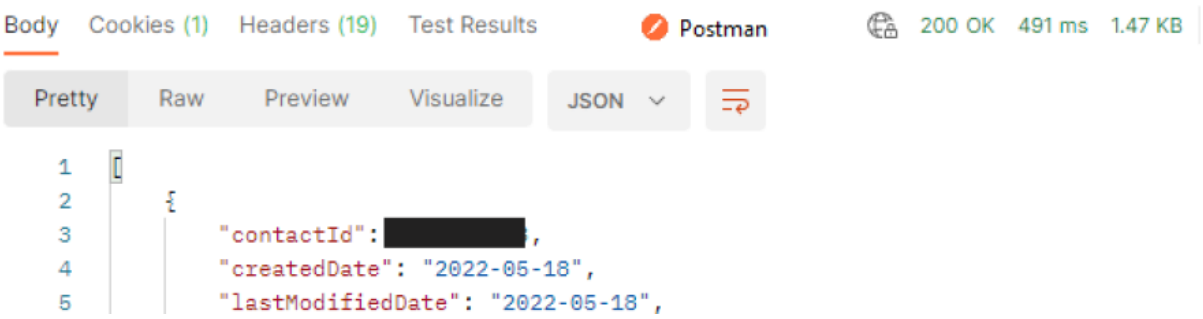
Figur 27: Tester - Kjøreresultater fra backend kodetester



Figur 28: Tester - Kjøreresultat fra route testing

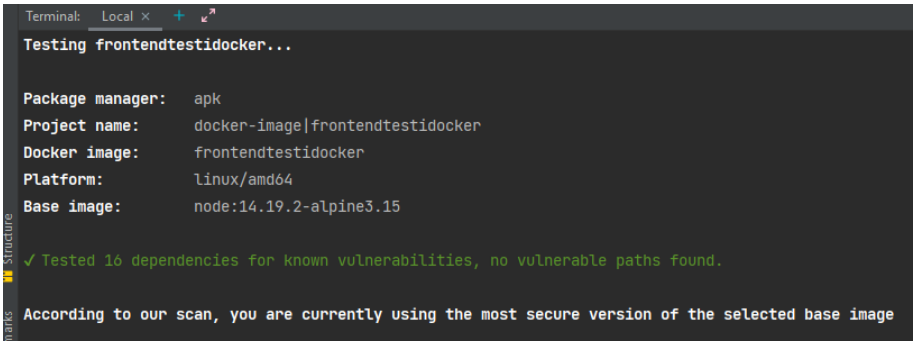
```python
def test_DB_create_kunde_exception_handler():
    with pytest.raises(HTTPException):
        create_customer(kunde=dummy_kunde_data_conversion, test=True, commit=False)
def test_ticket_creation():
    data = {...}
    new_ticket = Ticket(**data)
    assert new_ticket is not None

def test_ticket_attribute_created_right():
    ticket_data = {...}
    new_ticket = Ticket(**ticket_data)
    assert new_ticket.ticket_status is False
@pytest.mark.asyncio
async def test_router_ticket_gets_all():
    result = await get_complete_ticket()
    assert result is not None
```

Figur 29: Tester - Eksempel på testkode saksett fra forskjellige testfiler

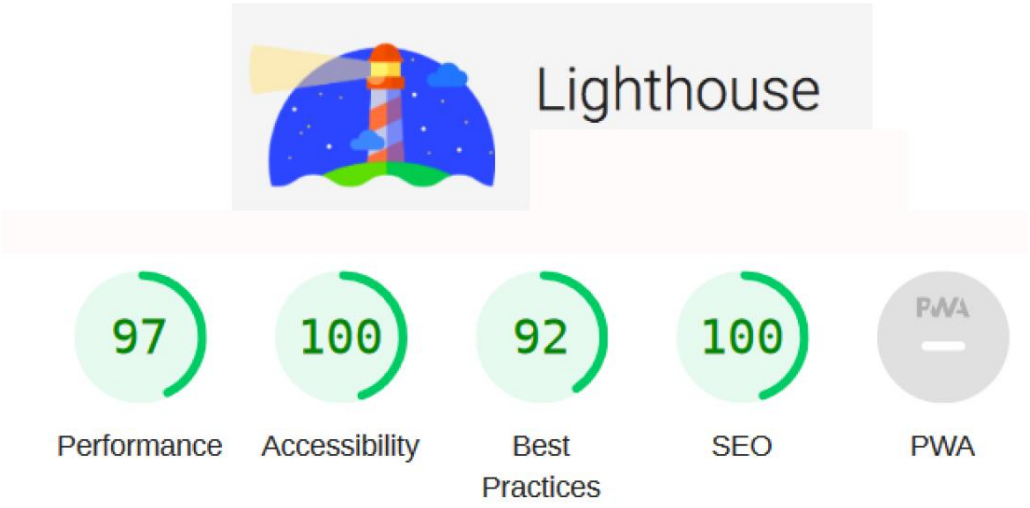


Figur 30: Tester - Kjøreresultat fra en test i Postman fra en integrasjonstest

Underliggende er en snyk test av frontend. For backend ligger testen i kapitelet om sikkerhet.



Figur 31: Tester - Snyk test av frontend

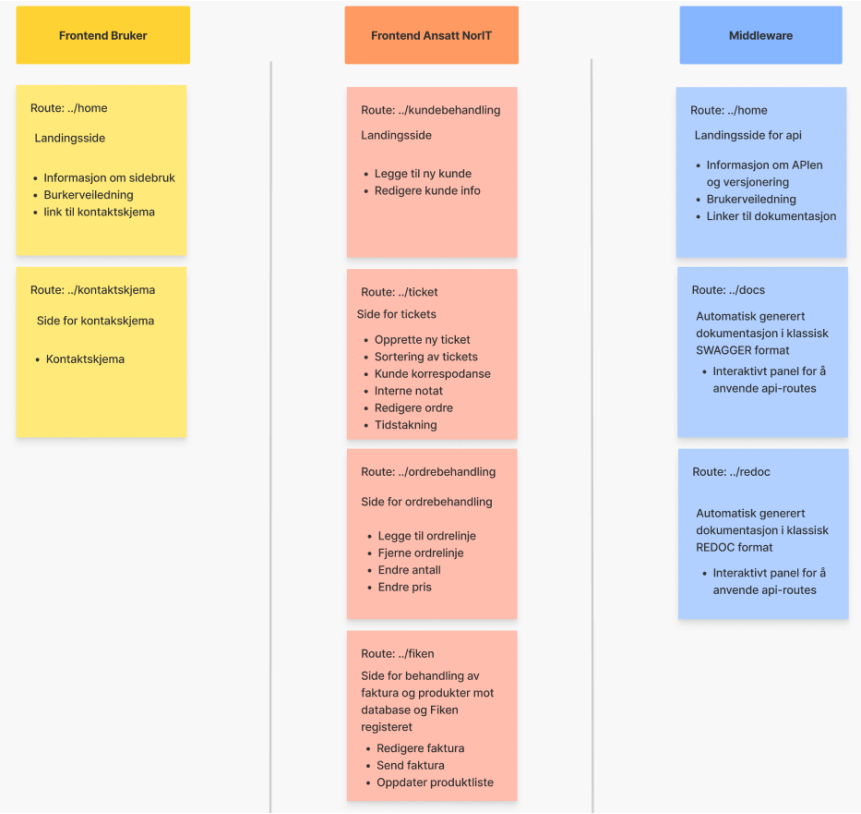


Figur 32: Tester - Tilgjengelighet og ytelsestest av frontend i Google Lighthouse

Vi har gjort kontinuerlige tester med Lighthouse som har gjort at vi forløpende har kunnet rette feilene våre underveis

14 Sideinnhold

Under er en figur med oversikt over de ulike sidene og hva de vil inneholde. Tickets og ordrebehandling har blitt slått sammen etter samtale med oppdragsgiver, da det ikke vil være noen ordre uten en ticket. Merk at kolonnen som er i gult, som viser «frontend bruker» er ikke en del av systemer vi produserer, men vi ønsket å ha de med for å gjøre det helhetlig og mer visuelt beskrivende.



Figur 33: Sideinnhold - Nor IT webapplikasjon

15 Teknisk løsning

I dette kapittelet presenteres systemkomponentene slik de er blitt implementert.

Generelt

Vi har gjennom prosjektet kontinuerlig vurdert mulige tekniske løsninger i samarbeid med oppdragsgiver. Vi har i stor grad hatt frihet til å velge hvilke teknologier vi har ønsket å bruke og på hvilken måte de benyttes. Med det sagt har vi prøvd etter beste evne å følge bransjetrender og oppdragsgivers ønsker når det kommer til skyklare løsninger og API first tankegang. Det igjen betyr at alle systemkomponentene kjøres i Docker og er klargjort for leveranse i skyen. Vi har valgt å blande konseptet middleware og backend slik at vi kan sørge

for at all trafikk til og fra frontend går gjennom samme punkt i systemet uten at vi har hatt behov for å produsere en ytterligere systemkomponent for å håndtere dette. Viktig her å påpeke at det er et unntak her, av sikkerhetshensyn har vi valgt å ikke gå for en middlewareløsning på Azure Active Directory autentisering.

Backend/Middleware/API

Denne delen er utviklet i Python med rammeverket FastAPI. Avhengigheter håndteres etter konvensjonell pip standard som vil si at vi har alle eksterne biblioteker listet i en requirements.txt fil og bruker pip til å installere disse i prosjektet. Vi har valgt å ikke benytte oss av andre avhengighetsstyringssystemer som f.eks. Poetry for å holde kompleksiteten i dette til et minimum. Denne delen av systemet leverer data til frontenden med JSON datastruktur over HTTP protokollen og har REST arkitektur. Autentiseringen her følger single sign-on prinsippet ved at man autentiseres et sted, i frontenden, får tilgang på en webtoken som systemet benytter seg av automatisk for bruk av backenden.

Frontend

Den er utviklet i JavaScript med React biblioteket som grunnlag for komponenter og struktur. Her benytter vi Yarn til å styre avhengigheter og det har vi valgt å gjøre til fordel for NPM da Yarn installerer avhengigheter parallelt mens NPM gjør det sekvensielt. Vi har valgt å ta vår smidige arbeidsprosess med over til brukergrensesnittet som tilbyr flere brukerstyrte tilpasninger ved justering av størrelser på arbeidsflater. Av tidshensyn har vi gjennom en kost-nytte-samtaler med oppdragsgiver kommet frem til at vi designet mobile first men kodet primært for desktop. Autentisering i denne systemkomponenten styres av Azure Active Directory

CDN

Vårt Content Delivery Network er i utgangspunktet kun implementert som et eksempel. Den leverer noe veldig grunnleggende CSS over HTTP protokollen som benyttes i på en velkomstside for API-et vår. Tanken her er at denne på sikt kan utvikles slik at den kan levere CSS på tvers av systemkomponenter slik at man kan garanterer et uniformt utrykk.

Annet

I tidlige prosjektsamtaler ble det uttalt at oppdragsgiver ønsket en SMTP integrasjon slik at deres kunder kan motta varsler o.l. Vi har dog valgt å gjøre en integrasjon mot Microsofts graph API for å styre dette. Det har vi valgt primært av to hensyn. Vi ønsket å holde backenden så nært en middleware som mulig, vi ønsket en situasjon der oppdragsgiver kan

gjøre et oppsett for mail i sin Azure klientportal og derfra styre hvilke informasjon en kunde får oppgitt når en mail sendes uten å måtte inn i koden og tilpasse konfigurasjonen av SMTP klienten. En klar ulempe med dette er at der hvor en SMTP ganske enkelt kan tilpasses ved et eventuelt plattformbytte, vil det være nødvendig med store kodeendringer for å tilpasse seg dette ved bruk av graph API-et. Denne ulempen kan forsvares ved at det er forholdsvis vanlig å tilpasse seg den skyplattformen man lever på og sånn sett er det forventet at det må større endringer til ved et bytte.

16 Fremtidige forbedringer og forandringer

Når det kommer til hvilke fremtidig endringer og forandringer som kan eller bør gjøres må det legges til at slike vurderinger best lar seg gjøre etter systemet har vært i drift i en periode. Av den grunn må det tas høyde for at våre vurderinger er mer teoretiske basert enn praktisk, økonomisk og behovsprøvd. Oppdragsgiver uttaler også at de forventer å vokse med ca. 2-3 kunder per mnd. Men at de ikke har satt seg noe tak for når de vil kutte i veksten. Med det utgangspunktet er det også vanskelig å si noe om en fremtidig løsning på skalerbarhet utover at det skal kunne skaleres. Vi velger derfor å se på løsninger som vil være hyperskalerbare heller enn moderat skalerbart.

Backend/middleware

Her er det naturlig at man på sikt bør skille de to. Etter hvert som antallet eksterne API integrasjoner øker vil et sammensatt system slik vi har per nå vokse seg frem til å bli et monolittisk system. Det er noe man ønsker å unngå spesielt med tanke på å holde applikasjonen i skyen. Man kan også gjøre en ny vurdering av verdien å splitte det som da blir middlewaren opp i mikrotjenester hvor hver av disse kan betjene hver sin eksterne API. Med det sagt er nok det viktigste å se på sikkerheten. Dagens løsning tilbyr en statisk webtoken tilgjengelig på frontenden for autentisering. Dette bør løses ved integrasjon av Azure Active Directory.

Frontend

På fremsiden er det vår mening at dette er et system som definitivt passer til en mikrotjeneste struktur med f.eks. Podium.Js som plattform. Det vil sikre at driftskritiske tjenester kan kjøre selv om andre elementer går ned. Slike elementer kan være kundekommunikasjon eller fakturering. Det er unødvendig at man ikke kan kommunisere med kunder fordi det er en feil i notatfunksjonen. En mikrotjeneste struktur sørger også for at man kan gjøre server side

rendering på enkelte elementer og client side rendering på elementer som krever det. Det igjen vil føre til en applikasjon som er både smidig og rask.

CDN

Denne komponenten kan bygges ut til å levere CSS og JavaScript komponenter til frontenden, backenden og til en ønsket mobilapplikasjon. Og vil da sørge for at man får samme uttrykk i alle applikasjoner på tvers av systemet. Man kan oppnå det samme ved å bruke Bootstrap men da med flere valgmuligheter og større behov for å beskrive konvensjoner og dokumentere benyttede valg.

Annet

Andre ting man kan vurdere er hvorvidt enkelte funksjoner i systemet kan flyttes og kjøres som serverless funksjoner i Azure. En slik funksjon kan være logging.

17 Refleksjon

Dette kapitlet skal vi gjøre refleksjoner over ulike valg som er tatt og gjort av oss igjennom de ulike prosessene av dette prosjektet.

17.1 Generelle refleksjoner

Risikoplan

Som utgangspunkt i risikoplan har vi nok vært litt naive i arbeidet med risikoplan og da spesielt i forbindelse med konsekvensanalysen. Vi har opplevd utslag i punktene som går på sykdom, varierende motivasjon, manglede disiplin og tidstyver i risikoplanen vår og spesielt med tanke på varierende motivasjon og tidstyver har vi sett at vår tiltro til lav konsekvens her har vært feil. Konsekvensen var satt som lav på disse punktene, men har vist seg å være høy. Dette er lærdom vi tar med oss og gjør en grundigere vurdering ved neste anledning.

Samarbeid og delansvar

Selv om samarbeidet generelt sett i gruppen, har vært bra er det fortsatt rom for forbedringer. Vi har hatt som avtale at man skal svare på meldinger, og være tilgjengelig for hverandre, samt at man skal gi beskjed dersom noe oppstår og man ikke får møtt til avtalt tid. Det har vi opplevd at ikke alltid har fungert slik vi hadde håpet det skulle, og vi har heller ikke klarte å løse det gjennom kommunikasjon. Stort sett stammer det fra forskjellige synspunkter hos medlemmer i gruppen. Dette er noe vi kunne tatt med veileder og fått hjelp til å løse, men det

har stort sett gått greit og ikke skapt for mye frustrasjon. Vi har også erfart at det har vært smått problematisk med samarbeidet for frontend og backend. Vi hadde avtalt en struktur hvor de som jobbet med frontend skulle sette sin egen backlog for oppgaver i prioritert rekkefølge, slik at de som jobbet med backend kunne synkronisere sin backlog og sørge for at frontenden fikk tilgang på det de behøvde til enhver tid. Dette var vi ikke gode nok på å følge opp og programmeringen ble haltende på grunn av det. Retrospektivt sett, burde vi definitivt ha hatt flere backlog-møter for å koordinere og sette felles backlogs.

Tidsestimater

Gjennom prosjektet har vi erfart at vi har truffet forholdsvis godt på estimatene for tidsbruk i de enkelte prosjektfasene. Det har gjort at vi har sluppet mye stressende arbeid inn mot leveringsfristen og gjør oss trygge på at vi kommer i mål til avtalt tid med oppdragsgiver. Estimatene våre, samt planleggingen av tidsbruken i forbindelse med andre krevende arbeider på universitetet ga oss tidlig en indikasjon på at vi oppgaven var av såpass kompleksitet at vi ikke ville kunne fullføre all kodingen før rapporten hadde leveringsfrist. Vi var derfor ute i god tid og avtalte med oppdragsgiver at koden først skulle være klar til overlevering inn mot presentasjonen i juni.

Tidlig oppstart og jevnlig arbeid med gode rutiner

Vi var tidlig i gang med prosjektet og fikk startet på planlegging i god tid før semesterstart. Resultatet av det var at vi kom godt i gang tidlig og det var en motiverende påvirkning og ga oss mye drivkraft. Vi har også hatt svært gode rutiner og struktur for når og hvordan vi jobber. Det har spart oss for mye ekstraarbeid i å avtale arbeidstider og møtetider. Alle har vært klar over når vi arbeider, hvilke plattformer vi benytter og hvordan vi kommuniserer. Jevnlig arbeid hvor vi kun har hatt avbrudd i forbindelse med eksamener og arbeidskrav i andre emner har sørget for at vi har kunnet opprettholde fremdriften i prosjektet fra januar til mai. Disse oppholdene ble avtalt så fort datoer lå klare på canvas og var godt forankret i gruppen.

Integrasjoner skyen og eksterne API-er

Etter det ble bestemt at applikasjonene våre skulle ligge på Azure og vi satt oss som mål å kjøre de som selvstendige webapplikasjoner ble valget om å bruke Docker enkelt da vi trodde det kom til å løse problemene våre med skyklarhet. Det viste seg dog at vi hadde en lekkasje i minnet på frontenden som Docker ikke kunne hjelpe oss med. Lekkasjen gjorde at Azures sikkerhetssystemet stoppet noe av funksjonaliteten i applikasjonen. For ordens skyld bør det

nevnes at vi var litt trege med å sette opp frontenden i Docker og legge den på Azure. Hadde vi vært tidligere ute med det kunne vi spart oss mye frustrasjon og omkoding. Underveis opplevde vi også en del krangling med API-et til Fiken. Vi har måttet registrere og bytte selskaper der et par ganger da gratislisensen bare varer en måned. Det kunne vi løst bedre ved å få Nor IT til å registrere et dummy-selskap og bare betalt 90kr for API tilgangen. Vi merket oss også at Fikens API bryter med beste praksis ved ikke å returnere referanser til opprettede objekter når de blir opprettet. Det førte til at vi for hver gang vi opprettet et objekt måtte loope gjennom alle tilsvarende objekter til vi fant riktig og fikk lokal lagring da Fiken opperer med egne unike id-er. Dette problemet kunne vi muligens løst ved å ta kontakt med Fiken for å høre om de ville kommet med en oppdatering i API-et. Vi burde definitivt brukt bedre tid på å sette oss inn i denne API-et når vi gjorde risikoanalyse.

Tidssprekk i utviklingsfaser

Med unntak av i det vi har kalt design fasen har vi truffet godt på våre estimater både når det gjelder hele faser, men også tidsbruk på individuelle oppgaver. Sprekken skyldes i hovedsak to ting. Først og fremst fordi vi valgte midtveis å splitte gruppen slik at vi kunne starte med implementering av designet i både kode og rapport. Den andre årsaken var at vi ble forholdsvis hardt rammet av Covid i denne perioden og flere av oss ble satt ut av spill over noe tid. Her kunne vi ha budsjettert med en annen team-sammensetning ettersom vi nok kunne sett for oss at vi kom til å dele gruppen delvis i denne perioden og da allokert tid et annet sted i prosjektet for å få et riktigere bildet på hvor tiden og ressursene ble benyttet.

17.2 Team refleksjon

Gruppen vår er satt sammen av mennesker som vi kjenner godt og vi har jobbet tett sammen under hele studieløpet. Dette har bidratt til at vi jobber godt sammen, kjenner hverandres sterke og svake sider, og kan sette arbeidskraften der hvor det er størst behov for dem. Vi har vært heldige med sammensetningen i gruppen vår, medlemmene er flinke, pliktoppfyllende og det ikke har vært noe krangling eller konflikter underveis i prosjektet. Støter vi på utfordringer eller uenigheter blir dette løst ved kommunikasjon og eventuelt en avstemning, hvor flertallet bestemmer. Vi mener å ha valgt en god strategi ved å ha møter hver morgen før vi skulle arbeide med bachelor oppgaven, på denne måten fikk vi en oversikt over hva vi alle jobbet med fra dag til dag, om vi trengte flere hoder for å løse et problem eller hvis noen sto fast på en deloppgave. Samtidig har det vært uker hvor motivasjonen ikke har vært på topp hos alle, men vi har da prøvd å motivere hverandre og hjelpe med å dra hverandre i gang på

litt tyngre dager, det skal ofte ikke mere til enn noen og samarbeide med for å kunne snu det. Vi har også hatt utfordringer når det kommer til teknologien, det har vært mye nytt å sette seg inn i, og hver ny teknologi har ofte flere teknologiske avhengigheter knyttet til seg. Så for å forstå en ny teknologi må man forstå flere. Dette er vanskelig å evaluere når det kommer til tids- og ressursbruk, men vi føler vi har klart å løse disse utfordringene på en helt ok måte. Den brede forståelsen man tilegner seg gjennom denne prosessen føles motiverende, og for hver teknologi man føler man mestrer, åpner det seg nye dører.

17.3 GDPR

I praksis vil det være mulig for Nor IT og slette og /eller anonymiseres kontaktpersonen i firmaer, men det kan ikke være mulig og slette hele firma fra systemet. Begrunnelsen for at det ikke kan være mulig å slette hele firma med alt innhold er fordi transaksjoner og primærdokumentasjon (innunder primærdokumentasjon går blant annet kunde- og leverandørspesifikasjon) må bli tatt vare på i minimum 5år med tanke på regnskap (Altinn, 2021).

18 Konklusjon

Prosessen vi har vært igjennom for denne Bachelor oppgaven har vært spennende, utfordrende og lærerik. Vi har fått muligheten til å benytte oss masse av den kunnskapen vi har tilegnet oss de siste tre årene, og nå opparbeidet oss nye erfaringer både teknisk, men også når det kommer til prosjektstyring, noe som vi håper å kunne benytte oss av når vi nå skal ut i arbeidslivet.

Olav har lagt få føringer for oss når det har kommet til designet av produktet, noe som har gitt oss frie tøyler og muligheten til både å utfolde oss kreativt, men også kjenne på ansvaret og tillitten han har gitt oss. Vi vil også nevne at koden til produktet vil ikke bli levert før i midten av juni, dette er etter avtale med oppdragsgiver.

Rapporten vi nå leverer og produktet vi kommer til å levere i juni er noe vi alle sammen er stolte av, og vi kan også se tilbake på prosjektstyringen og totalt sett si oss godt fornøyd med måten vi løste dette på sammen. Samarbeidet i gruppen har gått veldig fint under hele prosjektet, og kommunikasjonen innad i gruppen har vært god hele veien. Samarbeidet med oppdragsgiver og veileder har vært upåklagelig, vi har hatt møter med begge to enkeltvis, men også sammen. Vi har satt stor pris på deres gode innspill og tilbakemeldinger underveis i prosjektet. Tusen takk.

19 Referanser

Altinn (2021, 05. Mai). *Oppbevaring av regnskapsmateriale.* https://www.altinn.no/starte-og-drive/regnskap-og-revisjon/regnskap/oppbevaring-av-regnskapsmateriale/

Amazon. (u.å.). *What is AWS.* Hentet 26. April 2022 fra https://aws.amazon.com/what-is-aws/

Bangerter, J. (u.å.) *Dataflow Diagram Symbols, Types, and Tips*. Lucidchart. https://www.lucidchart.com/blog/data-flow-diagram-tutorial#dfd-levels

Barone, R. (2020, 11. September) *What are libraries in programming?.* Hentet 23. Februar 2022 fra https://www.idtech.com/blog/what-are-libraries-in-coding

Bbc.no.uk (u.å.). *Programming software and the IDE.* https://www.bbc.co.uk/bitesize/guides/zgmpr82/revision/4

Bilal, A. (2021, 3. Desember) *Introduction to API-First approach.* Hentet 23. Februar 2022 fra https://rapidapi.com/guides/api-first

code.visualstudio.com (u.å.). *Extension Marketplace.* Hentet 16. Februar 2022 fra https://code.visualstudio.com/docs/editor/extension-marketplace

Christensson, P. (2006). *SMTP Definition*. Hentet 10. Mai 2022 fra https://techterms.com/definition/smtp

dataled.academy (u.å.) *What are data types and why are they important?* Hentet 23. Februar 2022 fra https://dataled.academy/guides/data-types/

Datatilsynet (2018, 30. Oktober). *Iverksette styringssystem for informasjonssikkerhet*. https://www.datatilsynet.no/rettigheter-og-plikter/virksomhetenes-plikter/informasjonssikkerhet-internkontroll/etablere-internkontroll/iverksette-styringssystem-for-informasjonssikkerhet/

Design-lance. (2021, 4. Mars) *Mobile-First Design*. Design-lance. Hentet 16. Mai 2022 fra http://design-lance.com/tag/mobilefirst/

Digdir.no. (u.å.). *Universell utforming av ikt.* Hentet 14. Januar 2022 fra https://www.digdir.no/felleslosninger/universell-utforming-av-ikt/1499

Digdir.no. (u.å.). *WCAG 2.0 (Web Content Accessibility Guidelines)*. Hentet 14. Januar 2022 fra

https://www.digdir.no/felleslosninger/wcag-20-web-content-accessibility-guidelines/1738

Doshi, H (2016, 4. Desember) *The Three Pillars of Emprircism (Scrum).* Hentet 13.05.2022 fra https://www.scrum.org/resources/blog/three-pillars-empiricism-scrum?fbclid=IwAR17E06ZTcQ4GHuAKi8rucHJHsMEhOhJybE8S6fBjZ-iJIPkhUITtGh-MXc

Gisle, J. (2018, 30. November). *Personvernforordningen.* Store Norske Leksikon. https://snl.no/Personvernforordningen

Granevag, M. (2020, 31. Juli). *Backend.* Store Norske Leksikon. https://snl.no/backend

Granevag, M. (2020, 31. Juli). *Frontend.* Store Norske Leksikon. https://snl.no/frontend

Guilty.no (2019, 12. Mars) *Webapplikasjon eller nettside – hva er forskjellen?.* https://guilty.no/blogg/hva-er-forskjellen-p%C3%A5-en-webapplikasjon-og-en-nettside

Hamilton, T. (2021, 10. Desember). *Kanban Vs. Scrum: What's the Difference?* Guru99. Hentet 21. Januar 2022 fra https://www.guru99.com/scrum-vs-kanban.html

Hjelseth.com (2016, 3. August). *Hva betyr åpen kildekode (Open Source)?*

https://www.hjelseth.com/design/betyr-apen-kildekode-open-source/

i.ntnu.no (u.å.). *Risikovurdering.* Hentet 16. Februar 2022 fra https://i.ntnu.no/wiki/-/wiki/Norsk/Risikovurdering

JetBrains. (2022, 11. Januar). *PyCharm: Get started.* Hentet 1. Februar 2022 fra https://www.jetbrains.com/help/pycharm/quick-start-guide.html

Kristoffersen, B. (2020). *Databasesystemer* (5. Utgave). Universitetsforlaget.

Liang, M. (2021, 11. Mars). *Understanding Object-Relational Mapping: Pros, Cons, and Types.* Altexsoft. https://www.altexsoft.com/blog/object-relational-mapping/

Lucidchart (u.å.). *ER diagram symbols and meaning*. Lucidchart. Henter 13. Mai 2022 fra https://www.lucidchart.com/pages/ER-diagram-symbols-and-meaning

Marcia, S. G. (2021, 23. Juli) *Understanding REST APIs and their constraints*. Hentet 23. Februar 2022 fra https://www.webscrapingapi.com/rest-api-architecture-constraints/

Meiling, J (2019, 15. Januar) *Kanban in organizations Part ||*. Hentet 13.05.2022 fra

https://www.projectwizards.net/en/blog/2019/01/kanban-corepractices?fbclid=IwAR37_43nmjtRSHCFssExst9aow7BlRz1exG3gUvmPLEiEpOBI5vNipdbvXg#explicit-policies

Microsoft (u.å.). *Serverless Computing.* Microsoft Azure. Hentet 16. Mai 2022 fra https://azure.microsoft.com/en-us/overview/serverless-computing/

monocubed.com (2021, 6. September). *Typescript vs JavaScript: Why Choose Typescript Over JavaScript?*. Hentet 02.Februar 2022 fra https://www.monocubed.com/typescript-vs-javascript/

Norapong, M. (2021, 9. August) *Type Hinting – An Introduction.* Hentet 23. Februar 2022 fra

https://cpske.github.io/ISP/type-hints/introduction

Pabbi, T. (2018, 25. June) *Logging strategy for software applications*. Hentet 22. Februar 2022 fra https://codedesignetc.com/2018/06/25/logging-strategy-for-software-applications/

Posey, B. (2017, . September) *Computer exploit.* Hentet 23. Februar 2022 fra https://www.techtarget.com/searchsecurity/definition/exploit

Ranjan, R. (2021, 7. Oktober). *What is a framework in programming & why you should use one*. Hentet 16. Februar 2022 fra: https://www.netsolutions.com/insights/what-is-a-framework-in-programming/

React. (u.å.) *React A Javascript library for building user interfaces.* Hentet 15. Mars 2022 fra: https://reactjs.org/

Redhat (2020, 8. May) *What is a REST API?* Hentet 26. April 2022 fra https://www.redhat.com/en/topics/api/what-is-a-rest-api

Redhat (2018, 5. Januar) *What is CI/CD?* Hentet 23. Februar 2022 fra https://www.redhat.com/en/topics/devops/what-is-ci-cd

Owcarz, K. (2021, 2. September) *What is scrumban? | Definition + 5 Common Scrumban Myths*. DroidsOnRoids. Hentet 21. Januar 2022 fra: https://www.thedroidsonroids.com/blog/what-is-scrumban-definition-and-common-myths

Sandnes, E, F. (2018) *Universell utforming av IKT-systemer: Brukergrensesnitt for alle.* (2.utg.). Universitetsforlaget

Spring. (u.å.). *Building REST services with Spring.* Hentet 26. Januar 2022 fra https://spring.io/guides/tutorials/rest/

Skjørten, T. (2022, 24. Januar). *Hva er API? og 9 andre spørsmål og svar om API.* https://www.visma.no/blogg/hva-er-api-sporsmal-og-svar/

Stackoverflow. (2021, May). *Developer Survey*. Stackoverflow. https://insights.stackoverflow.com/survey/2021

Stephens, R. (2015). *Beginning Software Engineering.* John Wiley & Sons, Inc.

sysaid.com (u.å.). *What is a Ticketing System?* Hentet 23.Februar 2022 fra https://www.sysaid.com/resources/what-is-a-ticketing-system

Syed Hasan. (2022, 10. Mai). *Log4j RCE 101: An Overview for Beginner.* Gigasheet. https://www.gigasheet.co/post/beginners-log4j-rce101

Tafelski, M. (2022, 28. Januar) *What is Cloud Native?* Hentet 13.05.2022 fra https://www.trendmicro.com/en_no/devops/22/a/what-is-cloud-native.html?gclid=Cj0KCQjwg_iTBhDrARIsAD3Ib5gKMuiqI4onwz6qHsEK4dpmvLhArn96Gy4X7RnzqRmKxoDHntnSI0IaAry0EALw_wcB

Theastrologypage.com (2022). *Hva er et integrert utviklingsmiljø (ide)? Definisjonen fra techopedia – Utvikling – 2022.* https://no.theastrologypage.com/integrated-development-environment

Theastrologypage.com (2022). *Hva er skylagring? - Definisjon fra techoedia – Cloud-Computing – 2022.* https://no.theastrologypage.com/cloud-storage

TechTerms. (2011, 18. August). *Repository*. Hentet 22. Februar 2022 fra https://techterms.com/definition/repository

Uutilsynet.no. (u.å.). *WCAG 2.0-standarden.* Hentet 14. Januar 2022 fra https://www.uutilsynet.no/wcag-standarden/wcag-20-standarden/86

Visualstudio. (2021, 8. Desember). *Why did we build Visual Studio Code?* https://code.visualstudio.com/docs/editor/whyvscode

W3Schools. (u.å.) *SQL Injection* Hentet 20. Mai 2022 fra https://www.w3schools.com/sql/sql_injection.asp

20 VEDLEGG

20.1 Begrepsdefinisjoner

API

Et API er et programmeringsgrensesnitt som brukes for å sikre at flere tjenester/applikasjoner kan «snakke» sammen, og dermed utveksle informasjon. En API vil si noe om hvordan oppbygningen av et system er, og dermed hvilke krav som det andre systemet må ha på plass for å kunne opprette en kommunikasjon med et annet system (Skjørten, 2022).

AWS

AWS forkortet for Amazon Web Services er skyplattform levert av Amazon. De tilbyr over 200 tjenester i form av infrastrukturteknologier som databehandling, lagring og et vidt utvalg av databaser spesialbygd for forskjellige applikasjoner. Også nyere teknologier som maskinlæring og kunstig intelligens er noe de har fokus på (Amazon, u. å.).

Backend

Backend tar for seg den delen av programvaren som ligger nærmest databasen hvor dataene blir lagret. Dette er den delen av funksjonalitet i et program som brukeren ikke ser eller nødvendigvis har noe forhold til (Granevag, 2020).

Bibliotek(kodebibliotek)

Et bibliotek i kodesammenheng er i korte trekk en kolleksjon av ferdigskrevet kode som brukere av bibliotekene kan ta i bruk for å optimalisere oppgaver i eget prosjekt, et eksempel på et slikt bibliotek er NumPy, som er en kolleksjon brukt for å lage større og mer tolerable lister (arrays) ofte brukt i sammenheng med AI, machine learning og deep learning (Barone, 2020).

CI og CD

CI står for Continuous Integration og er en automatisert prosess, integrert på riktig måte vil CI-kode forandringer til en applikasjon eller ett prosjekt testes og deretter bli sendt til et felles repository. CI er en løsning til det kjente problemet, hvor en har for mange «branches» i repositoriet sitt, hvor implementering av kode ofte vil møte på konflikter.
CD står for Continuous Delivery eller Deployment og inneholder mange av de samme konseptene som CI. En av hovedegenskapene til CD er prosessen hvor en utviklers kode går fra repositoret til produksjonen (ut til kundene). CD sin hovedoppgave er å gjøre denne jobben enklere for utvikleren(e) (redhat, 2018).

Cloud Storage også kalt Skylagring

Dette er lagring av data på eksterne servere, som en kan ha tilgang til fra internett. Ofte med veldig god sikkerhet som man vil ha tilgang til fra hvilken som helst datamaskin, eller mobilenhet. Serveren både drives og administreres som oftest av en ekstern leverandør av skylagringstjenester (theastrologypage.com, 2022).

Cloud native

Cloud native er applikasjoner som kjører i og er laget for skyen. Utviklere har med tanke på skyen optimalisert og bygget infrastrukturen til applikasjonen slik at den skal kunne kjøres derfra uten problemer. Det er fem hovedkomponenter konteinere, microserviser, service meshes (altså at applikasjon er delt opp i flere lag slik at den enklere kan skaleres), infrastruktur og API-er (Tafelski, 2022).

Code completion

Oversettes til autofullføring eller kodefullføring. Dette er for at man skal kunne spare litt tid mens man skriver kode. Når hen begynner å skrive den første delen av en funksjon, vil det bli foreslått eller fullført den funksjonen hen har startet på (bbc.co.uk. u.å.)

Datatyper

Datatyper sier noe om hvordan dataene blir lest eller lagret på. For eksempel i en database er det vanlig at verdiene ligger inne med en datatype som sier noe om hvordan type data som er lagret der, og hvilken data som kan bli lagret der. Datatype skal fortelle et system hvordan den skal lese å forstå det som ligger der. Noen eksempler på datatyper er: Integer, denne tar for seg heltall. String, denne datatypen tar for seg tekst. Boolean, denne representerer «true or false», enten er den verdien sann eller ikke (Dataled.academy, u.å.).

Exploit

Exploit er et angrep på et datasystem, hovedmålet til en exploit er og overta spesifikke deler av et system. Exploit angriper gjerne den svakere delen av systemet, applikasjonen eller koden. Måten en utvikler kan forhindre en exploit er ved å gjennomføre en kode fiks eller kjøre ut en oppdatering av systemet til brukerne sine. Et eksempel på exploit angrep kan være Structured Query Language (SQL)-injection, hvor vedkommede prøver å ta seg inn i en web applikasjon gjennom dens database (Posey, 2017).

Frontend

Det visuelle i et program som brukeren samhandler direkte med. Koblingen mellom backend og frontend kan for eks. være en «lagre»-knapp. Så når brukeren trykker på den vil sette i gang lagringsprossessen til databasen i backend (Granevag, 2020).

IDE (Integrated Development Environment)

Eller Integrert utviklingsmiljø som det kalles på Norsk vil være en applikasjon eller programvare som er basert på en «arbeidsbenk» som er designet for å hjelpe en utvikler med å bygge kode i for å lage en ny programvare (theastrologypage.com, 2022).

Log4jrce

Log4jrce er en sammenslåing av "Log4j" og "rce". Log4j er et verktøy/bibliotek for loggføring skrevet i Java og er i vidstrakt bruk i mange applikasjoner. Det ble i nyere tid oppdaget et sikkerhetshull i dette verktøyet som tillater en angriper å utføre "Remote Code Execution" eller "rce" på en sårbar maskin. Kort fortalt utnytter angripere en sårbarhet i måten Log4j loggfører informasjon på, hvor de tvinger offerets maskin til å kjøre skadelig programkode (Syed Hasan, 2022).

Logging

Logging er en prosess i systemet som skriver til et sekundært grensesnitt til bruk i feilsøking. Det er mange forskjellige muligheter med logging. Logging kan registrere nesten alt som skjer i systemet, men vanligvis konfigurerer du den til å logge det du vil se og har bruk for. Noen av de populære tingene utviklere bruker logging til er når de sjekker for eksempel ting som kan være feil, advarsler, informasjon og feilsøking for å se for eksempel hva og hvor noe i koden gikk galt (Pabbi, 2018).

Object-Relational Mapping

Object-relational mapping gir et objekt orientert lag mellom relasjonsdatabaser og objektorientert programmeringsspråk for å slippe å bruke SQL-spørringer. Det gjør det mulig med objekt orienterte språk som f.eks. Java, C# å kommunisere med databasen ved å konvertere språket så databasen forstår språket (Liang, 2021).

Open-Source

Eller åpen kildekode som det kalles på Norsk vil tilsi at kildekoden som er benyttet for å skape et program, hvorpå denne kildekoden eller forkortet kalt kode er publisert og brukerne kan modifisere, og distribuere koden som de selv ønsker (hjelseth.com, 2016).

Rammeverk

Et rammeverk i programmering er et verktøy som legger til rette allerede underliggende ferdig komponenter eller løsninger, gjerne i form av programmer, biblioteker og kompilatorer. Det kan sammenlignes som et fundament eller grunnmuren i et hus som gjør det enklere å begynne å bygge huset med engang uten å måtte begynne fra start (Ranjan, 2021).

REST

REST eller RESTful står for "REpresentional State Transfer" og er en systemarkitekturs form for hvordan tjener og server samhandler i form av et sett arkitektoniske begrensninger. Når en klientforespørsel forekommer via et RESTful API vil representasjonen av ressursens tilstand sendes til rekvirenten via http ved bruk av et format enten i JSON, HTML, XLT, PHP, Python eller ren tekst (RedHat, 2020). Ideen bak å benytte REST er pålegge regler på API-et får å få mer ytelse, enkelhet, synlighet, skalerbarhet, portabilitet, modifiserbarhet og pålitelighet (Marcia, 2021).

Repositories/Repository

Et repository eller «oppbevaringssted» er en sentral fillagringsplass som brukes i programvareutvikling. Den brukes til å lagre flere versjoner av filer og brukes som en versjonskontroll. Det hjelper utviklere ved å tilby en måte å lagre filer og holde dem strukturerte på. Fillaggringsplassene er ofte lagret på en server slik at flere personer kan jobbe mot de samme filene, men kan også brukes på en lokal maskin for én enkelt bruker (TechTerms, 2011).

Serverless

Ved serverless databehandling kan utviklere bygge applikasjonen uten å måtte administrere infrastrukturen selv. Med serverless er det valgte skytjenesteleverandør som automatisk administrerer og skalerer infrastrukturen som kreves for å kjøre koden. Det er fortsatt servere som kjører koden, men serverless navnet kommer fra at oppgavene knyttet til infrakstrukturen og administrasjon blir usynlig for utviklerne i for eksempel en bedrift som måtte ta i bruk en tredjepart ved å bruke serverless for å sende og kjøre koden (Microsoft, u.å.).

SMTP

SMTP er en forkortelse for "Simple Mail Transfer Protocol" og er i korte drag en protokoll som er brukt for å sende eposter over internett. Det fungerer slik at SMPT sender en beskjed

til en server for eposter, serveren bruker videre SMTP teknologien til å videre sende beskjeden til riktig epost server. Dette er en automatisert prosess og den sørger for at utsenderen av eposten får sendt informasjonen hvor den skal (TechTerms, 2006).

SQL

SQL er forkortelse for "Structured query language", eller på norsk: strukturert spørrespråk. Kristoffersen (2020, s. 4) definerer SQL som et språk spesielt konstruert for å jobbe mot databaser. Ved å bruke SQL kan du manipulere databaser. Du kan for eksempel gjøre spørringer for å hente spesifikke data fra en database du ønsker, slette data, legge til data, endre data eller lage nye tabeller for å sette inn data systematisk.

SQL Injection

En av de mest vanlige formene for datangrep i dag er gjennom SQL injections, Det er et angrep mot en SQL server som har til hensikt og ødelegge og påføre skade i system eller database. Hackere eller inntrengere i systemet legger inn queries som SQL serveren kjører hvis de ikke blir stoppet, derfor er det viktig å ha kontroll på alt som går inn i SQL serveren og databasen (W3Schools, u.å.).

Typehinting

Typehinting er en måte å kode på utvikleren kan bruke for å øke lesbarheten i kode under utvikling for mennesker og datamaskiner, typehinting gir en bedre kodelesbarhet, hjelper med å redusere feil i kode i større prosjekter og kan sees på som en type virkelighetstids dokumentasjon i ett prosjekts livssyklus (Norapong, 2021).

Utvidelser

I for eksempel programmet Visual Studio Code, som er et IDE (Integrated Development Environment), kan man legge til utvidelser som lar deg legge til andre språk, debuggere og andre verktøy som du enkelt kan installere for å gjøre arbeidsprosessen enklere for programmerere (code.visualstudio.com, u.å.).

Webapplikasjon

En webapplikasjon vil være en dynamisk nettside hvor det kreves autentisering eller innlogging med bakgrunn i at innholdet på siden vil kunne endre seg ut ifra hvilke tilganger den enkelte brukeren har rettigheter til å se. I tillegg til hva du har rettigheter til å se, vil det også kunne være muligheter for brukeren selv til å gjøre endringer eller generere innhold til webapplikasjonen (guilty.no, 2019).

20.2 Gruppe kontrakt

Denne kontrakten gjelder for gruppe MH^2EK^2 og dets gjennomførsel og arbeidsprosess i bacheloroppgaven for Nor IT AS. Kontraktforandringer kan kun gjøres ved at gruppen er enstemmig i en forandring i kontrakten.

Målet med denne bacheloroppgaven vil være:

1. Gruppen skal lage en PSA-applikasjon for Nor IT AS.
2. Vi ønsker å sette høye krav til oss selv og utførelsen av denne oppgaven, og er innforstått med at det vil kreve mye arbeid av hvert enkelt medlem.
3. Anvende tilnærmet kunnskap fra foregående semestre på best mulig måte for å løse oppgaven gitt til oss.
4. Bistå hverandre på best mulig måte og ha en god gjennomgående dialog under hele prosjektet slik at alle føler seg inkludert og ivaretatt under hele prosessen. Dette vil på sikt sikre oss i større grad at vi får et sluttprodukt vi er stolte av å kunne vise frem, samt at et positivt arbeidsmiljø ser vi på som et vesentlig element i et hvert prosjekt.

Kjernetid for å arbeide med bachelor oppgaven vil være:

Dag	Tid
Tirsdag	10:00 – 16:00
Onsdag	10:00 – 16:00

Regler:

1. Møte presist til avtalt tidspunkt. Ved gjentatte brudd må vedkommende kjøpe en valgfri snacks til 50kr til de andre gruppemedlemmene.
2. Skulle man bli stående fast i en oppgave eller det skulle oppstå forsinkelser må dette kommuniseres.
3. Taushetsplikt gjelder både for informasjonen vi får fra Nor IT, for det generelle arbeidet og privatinformasjon.
4. Vi skal selvsagt både ha respekt og ta vare på hverandre.
5. Produsert materiale skal ha sikkerhetskopi via Microsoft Teams, Global Information Tracker (GIT), Dropbox eller lignende.
6. Det vil bli gjennomført demokratisk avstemning ved uenighet.

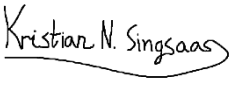
Houssam J. Melli

Henrik B. Hansen

Halvor U. Blindheim

Elin Handeland

Ketil Ustad

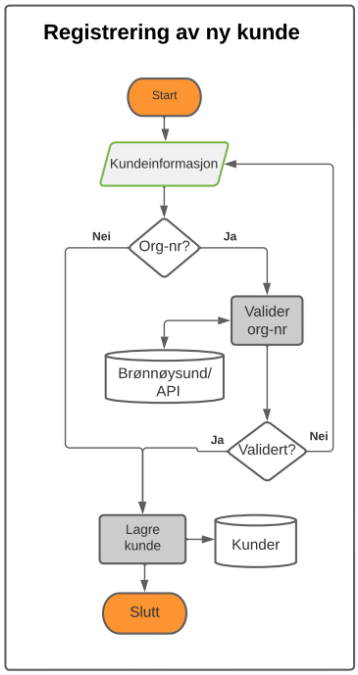


Kristian N. Singsaas

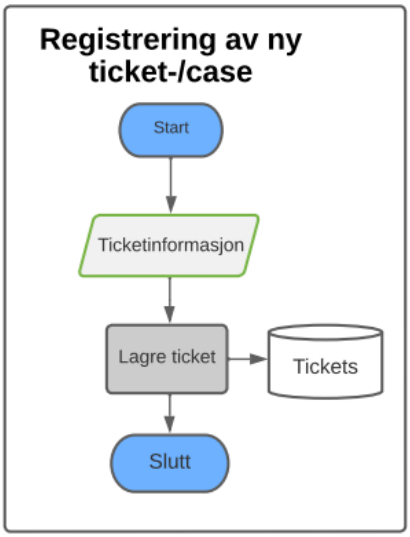
20.3 Diagrammer

DFD diagram

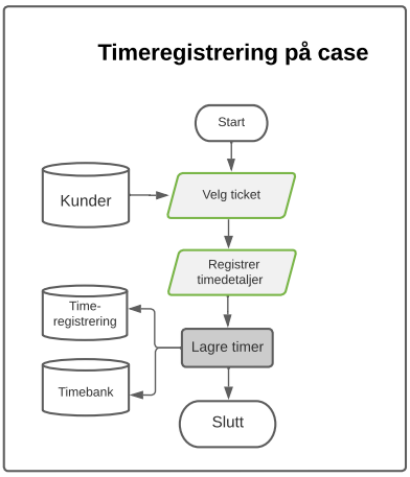
Figurene under er dataflyt diagrammer som representerer et utvalg funksjoner og aktiviteter i applikasjonen.



Figur 34: DFD level 1 - Registrere kunde



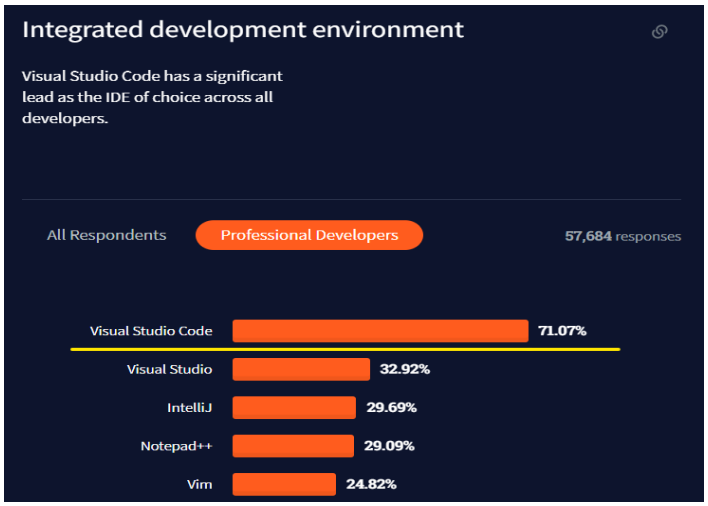
Figur 35: DFD level 1 - Registere ny ticket/case



Figur 36: DFD level 2 - Timeregistrering på case

Mest brukte utviklingsmiljø

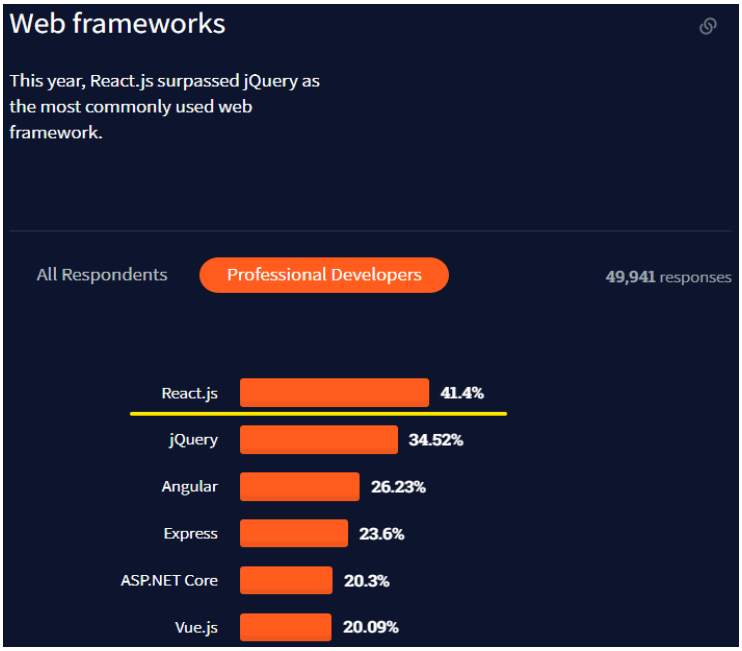
Figuren under illustrer de mest populære utviklingsmiljøene i 2021 gjort av Stackoverflow.com gjennom en spørreundersøkelse (Stackoverflow, 2021).



Figur 37: Mest populære integrated development environment, 2021, av Stackoverflow. https://insights.stackoverflow.com/survey/2021#integrated-development-environment

Mest utbredte teknologi for utvikling av webapplikasjoner

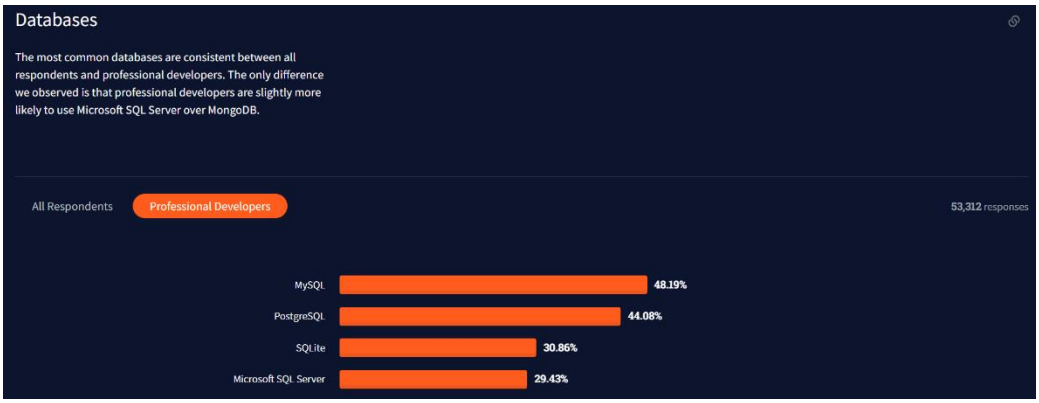
Figuren under illustrer de mest utbredte teknologiene for utvikling av webapplikasjoner i 2021 gjort av Stackoverflow.com gjennom en spørreundersøkelse (Stackoverflow, 2021).



Figur 38: Mest populære web rammeverk, 2021, av Stackoverflow. https://insights.stackoverflow.com/survey/2021#most-popular-technologies-webframe-prof

Mest brukte databaser

Figuren under illustrer de mest brukte databasene i 2021 av respondenter og profesjonelle utviklere gjort av Stackoverflow.com gjennom en spørreundersøkelse (Stackoverflow, 2021).

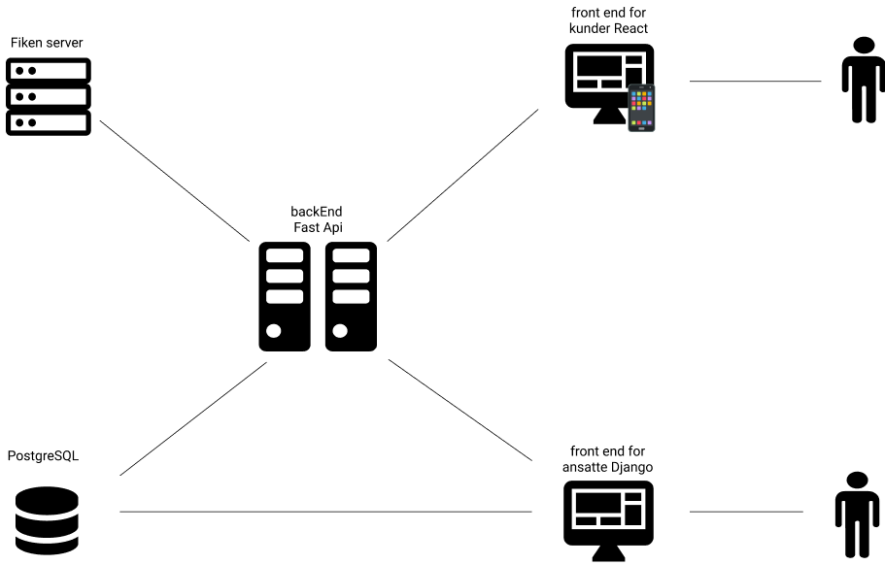


Figur 39: Mest populære databaser, 2021, av Stackoverflow. https://insights.stackoverflow.com/survey/2021#most-popular-technologies-database-prof

20.4 Utgåtte figurer og diagrammer

Systemarkitektur

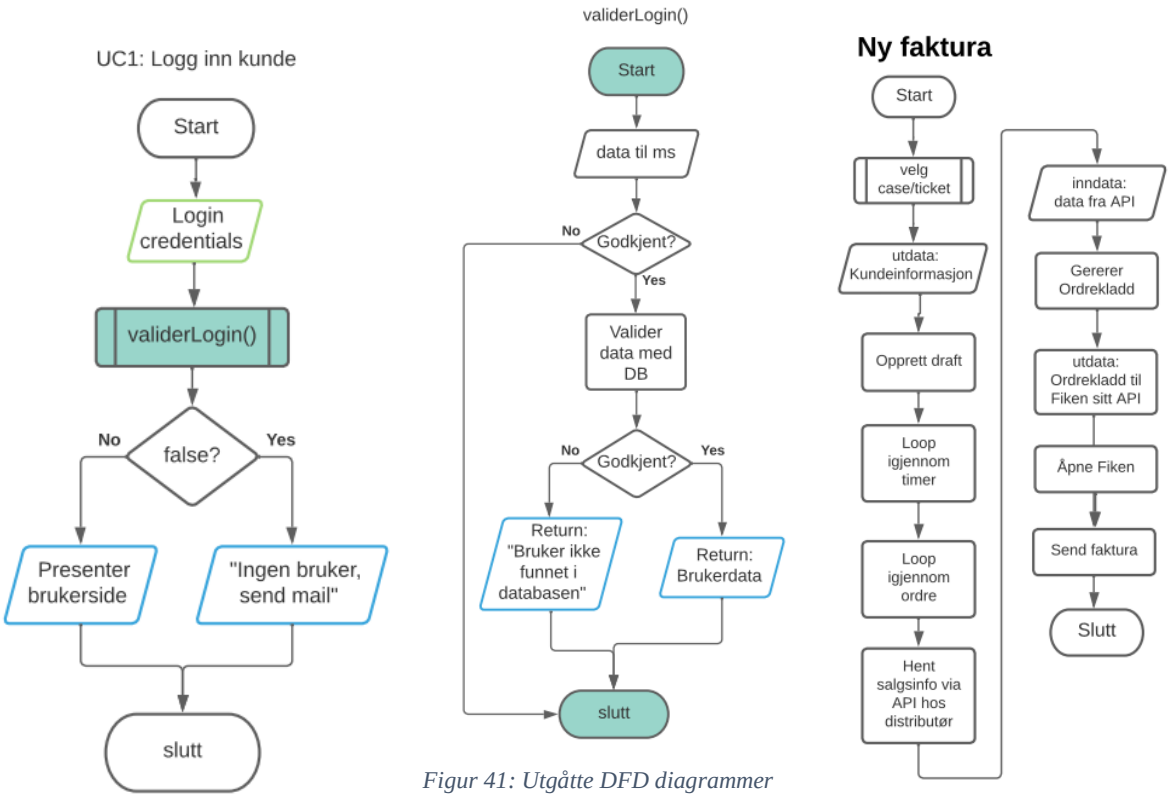
Ettersom kunde ikke lenger vil ha direkte tilkobling eller logg inn muligheter til webapplikasjonen har systemarkitekturen blitt revidert fra bildet under. Vi fant også ut at vi har behov for flere eksterne API'er, ett i Brønnøysund registeret for å hente ut nødvendig data. Microsoft Graph API for å benytte oss av Microsoft sine Cloud service ressurser, hvor vi integrerer et kommunikasjonsvindu i webapplikasjonen hvor ansatte kan kommunisere med kundene.



Figur 40: Utgått systemarkitektur diagram for Not IT webapplikasjon - Første utkast

Dataflytdiagrammer

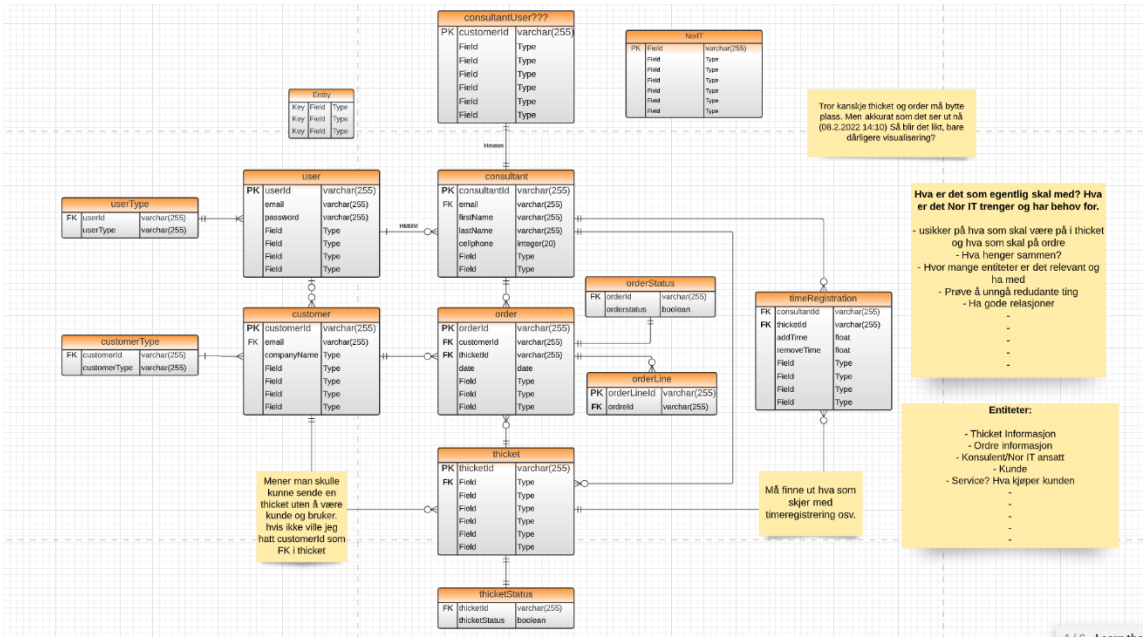
Disse flytdiagrammene som er presentert under representerer et par funksjoner som er blitt forkastet i løpet av prosessen. De to første ble foreldet etter vi implementerte Microsoft AD-login noe som resulterte i at funksjonene ble overflødige da Microsoft tar seg av login. Etter at vi lærte mer om Fiken og for eksempel API hos distributør, ble "Ny faktura" foreldet da den er mangelfull med tanke på de tekniske aspektene ved oppgaven som skal løses.



Figur 41: Utgåtte DFD diagrammer

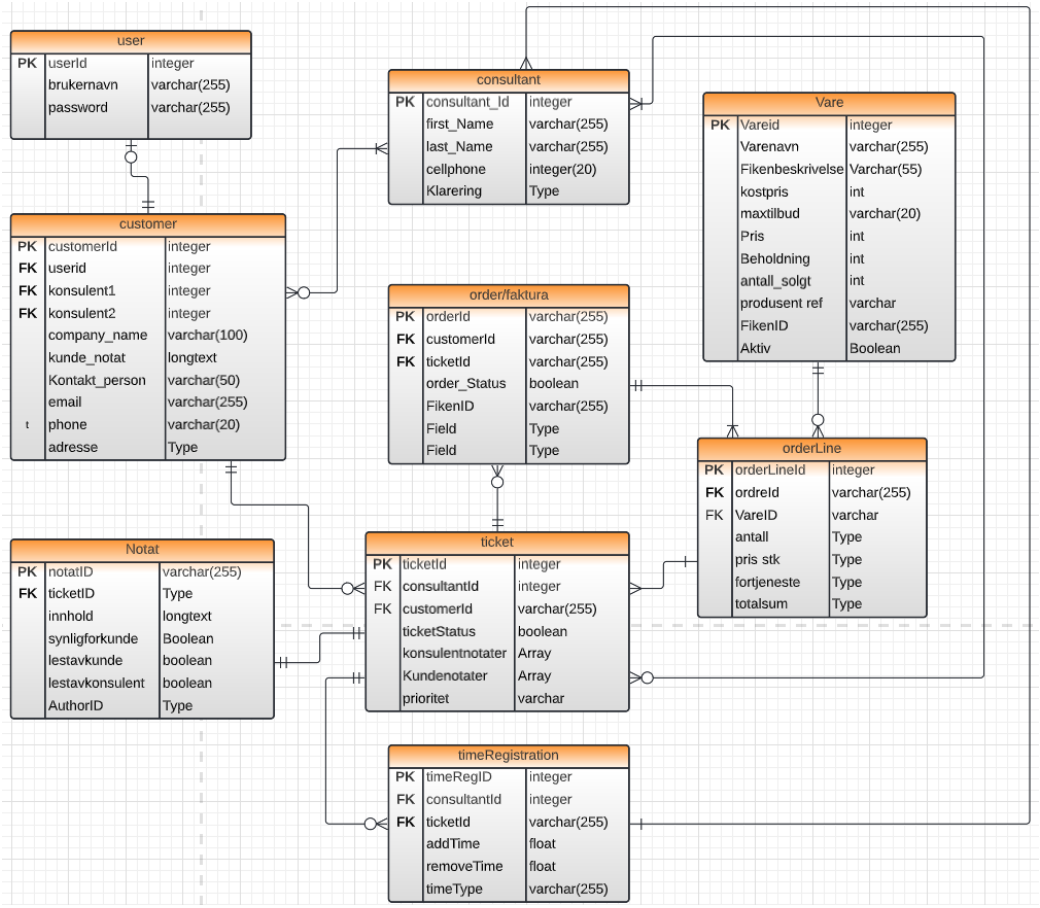
Databasemodell

Under ligger det tre versjoner av utgåtte databasemodell-diagrammer som fra første til gjeldende reviderte utkast som ligger i selve rapporten er blitt gjort om på flere ganger. Første utkast er den første kladden hvor vi benyttet oss av tanke/ide-kart for å kartlegge hva vi hadde behov for i webapplikasjonen på et tidlig stadium. Både første og andre utkast tar for seg hvordan vi så for oss hvordan databasen kunne blitt seende ut dersom kundepålogging fortsatt var et faktum. Tredje utkast minner mer om, enn de to andre vår siste versjon og viser hvordan alt henger sammen og hvordan fremgangsmåten vår var når vi skulle opprette et script som setter opp databasen. Databasemodellen har sett mange forandringer gjennom arbeidet vårt ettersom behov har dukket opp, eller at vi har funnet ut at noe er blitt redundant eller utgått. Det kan også med stor sannsynlighet tenkes at i fremtiden hvis webapplikasjonen blir jobbet og utviklet videre på at nye behov vil dukke opp og at modellen må revideres videre på, for eksempel hvis kunde logg inn skulle blitt implementert.

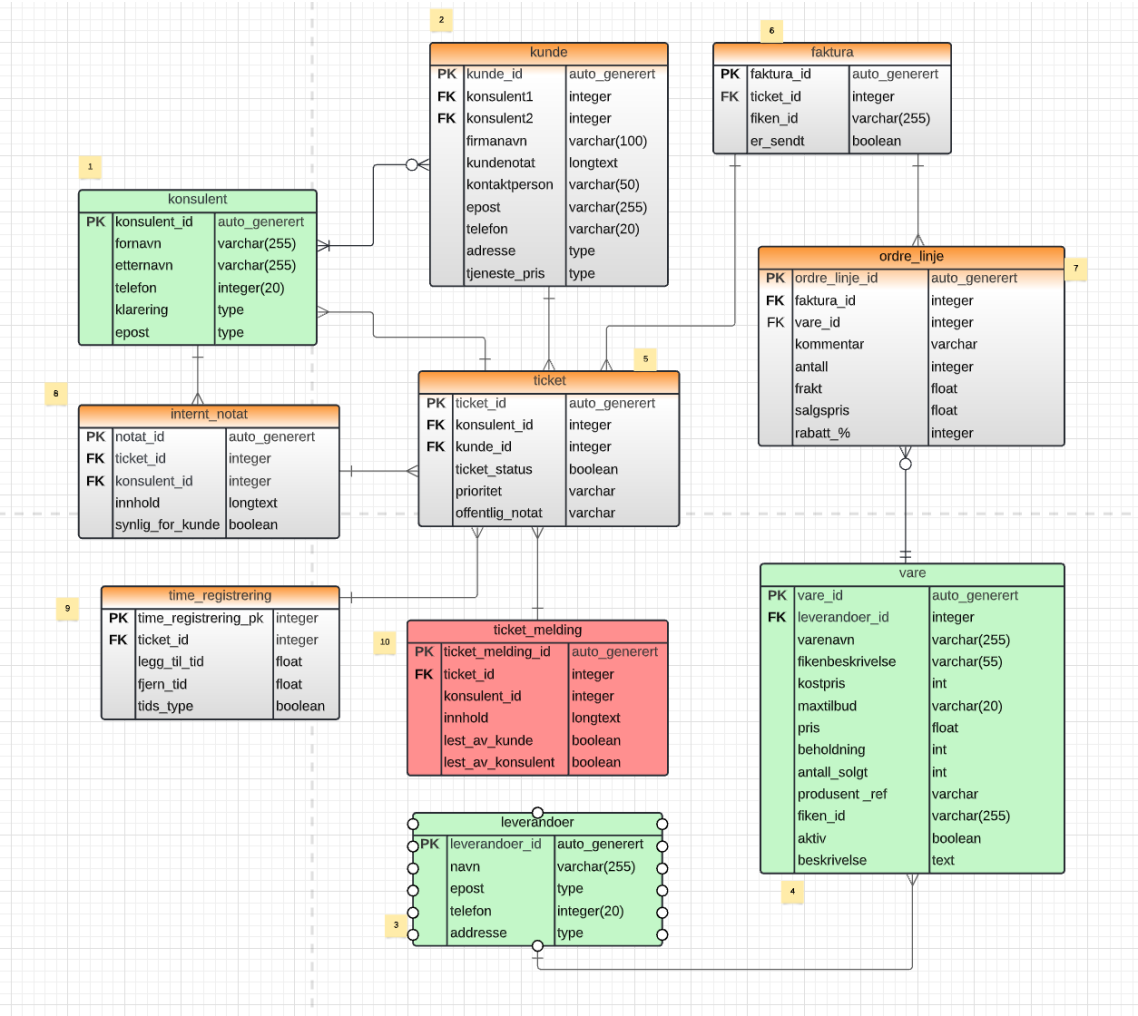


Figur 42: Utgåtte databasemodeller

(*Utgått databasemodell diagram for Nor IT webapplikasjon – Første Utkast)*



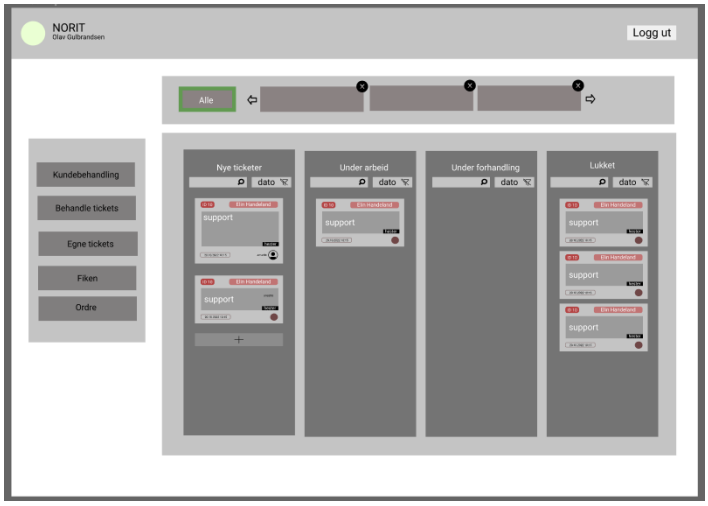
(Utgått databasemodell diagram for Nor IT webapplikasjon – andre Utkast)



(Utgått databasemodell diagram for Nor IT webapplikasjon – Tredje utkast)

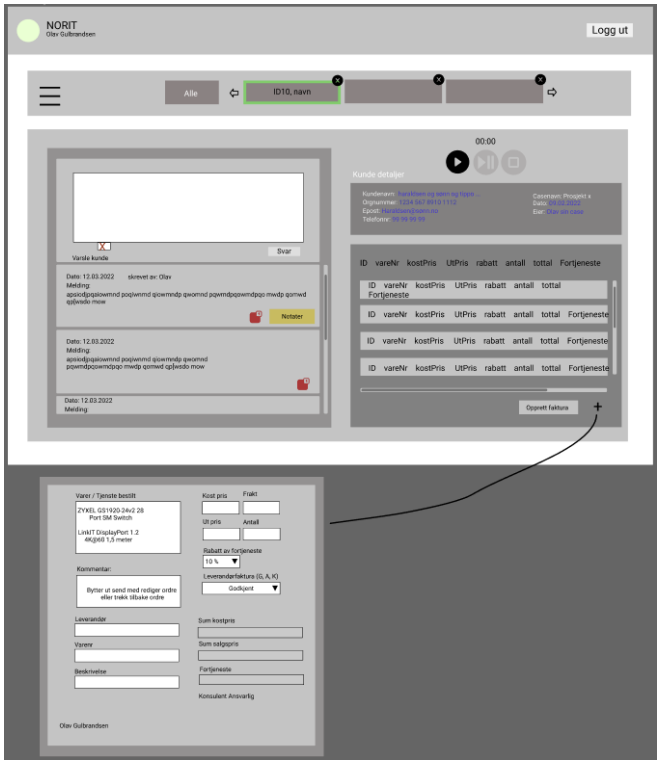
Wireframes

Fra det første utkastet av wireframes hadde vi en litt annen tilnærming til systemet enn hva det endelige ble. Her hadde vi en ide om å hente inspirasjon fra Kanban tavler, slike som man for eksempel kan se i Trello. Dette ville kunne vært en god ide, men vil nok være mer passende for bedrifter som har færre caser i løpet av en dag enn det Nor IT har. Bildet under viser en oversikt over alle casene bedriften har i systemet.



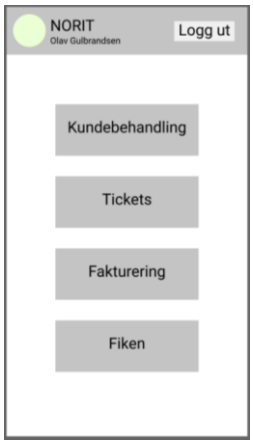
Figur 43: Utgåtte wireframes

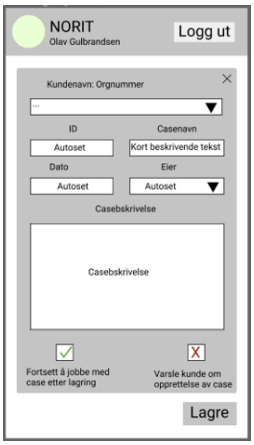
Bildet under viser hvordan systemet ville ha sett ut når en case er åpnet. I vinduet til venstre er notater, og til høyre ville man sett ordrelinjene. Pop up'n under beskriver hvordan det kunne bli lagt til nye varelinjer i ordren.



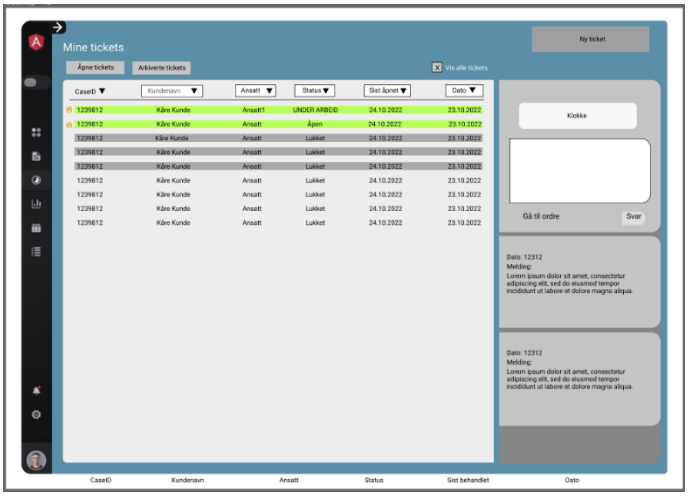
I starten av prosjektet hadde vi en tanke om at de ansatte skulle kunne bruke hele systemet på mobil, og vi laget noen utkast til hvordan det ville se ut. Etter et møte med Olav fikk vi avklart at det aller viktigste for de ansatte var og kunne bruke klokken, for å starte og stoppe tid når de er ute hos kunder. De andre funksjonene vil i svært liten grad bli benyttet. Under kan du se wireframes av mobilsiden vår, slik de var i første utkastet. Fra venstre side kommer

først et bilde av forsiden, videre kommer muligheten for å opprette ny case, en oversikt over alle tickets, og til slutt hvordan en åpne ticket ville kunne ha sett ut.





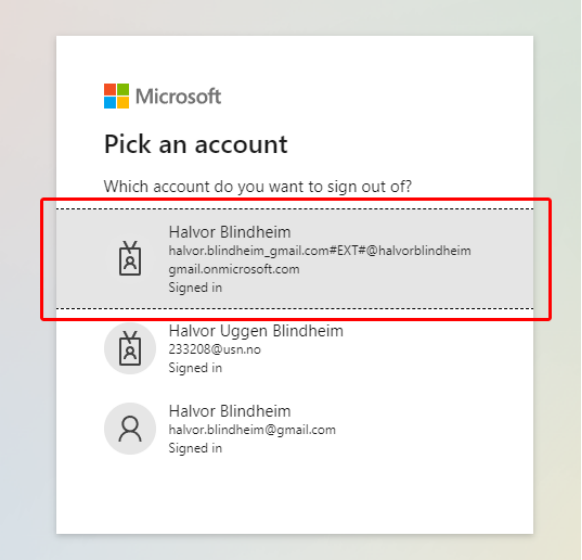
Under kan du se det andre utkastet av wireframe's, her har vi tatt en tilnærming mot hvordan en forenklet mailkonto ser ut og hentet inspirasjon derfra. Det nærmer seg nå det endelige designet. Siste utgave av wireframe kan du se under wireframe i selve rapporten.



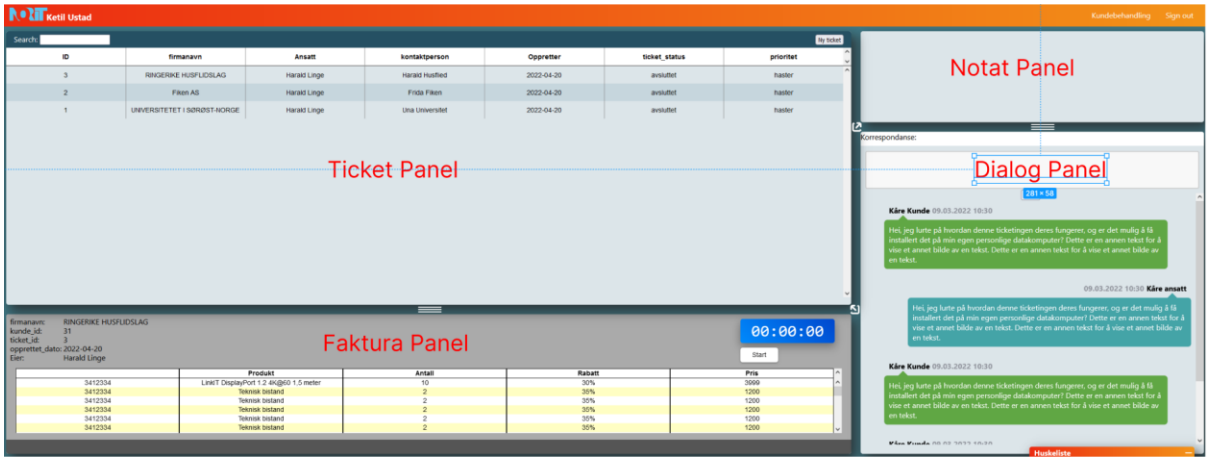
20.5 Brukerveiledning

Siden vår applikasjon ikke er fullstendig utviklet enda, vil bildene vise hvordan applikasjonen fungerer til nå. Der funksjonene er ufullstendige er det referert til bilder av prototypen.

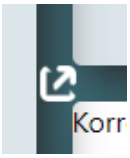
Når man først åpner applikasjonen via en nettleser, får man beskjed om å logge seg inn via sin Microsoft-konto.



Når man har logget seg inn, vil man bli presentert med hovedsiden på applikasjonen, som er delt inn i 4 vinduer/paneler. Oppe til venstre har man en oversikt over alle kundeforhold med tickets, og under det er presentasjon av eventuell faktura som er knyttet til den ticketen man har valgt å jobbe på. Oppe til høyre er vinduet for notater knyttet til en ticket og i vinduet nede til høyre finner man mail-korrespondanse mellom kunde og Nor IT.



Om man har behov for å modifisere størrelsesforholdene på disse vinduene, kan man gjøre det ved å benytte seg av knappene som er posisjonert mellom panelene. Disse fungerer på samme måte som ved skalering av et vindu i Windows og størrelsen man velger lagres til neste gang man starter applikasjonen.



Denne skalerer i alle retninger.

Denne skalerer kun vertikalt

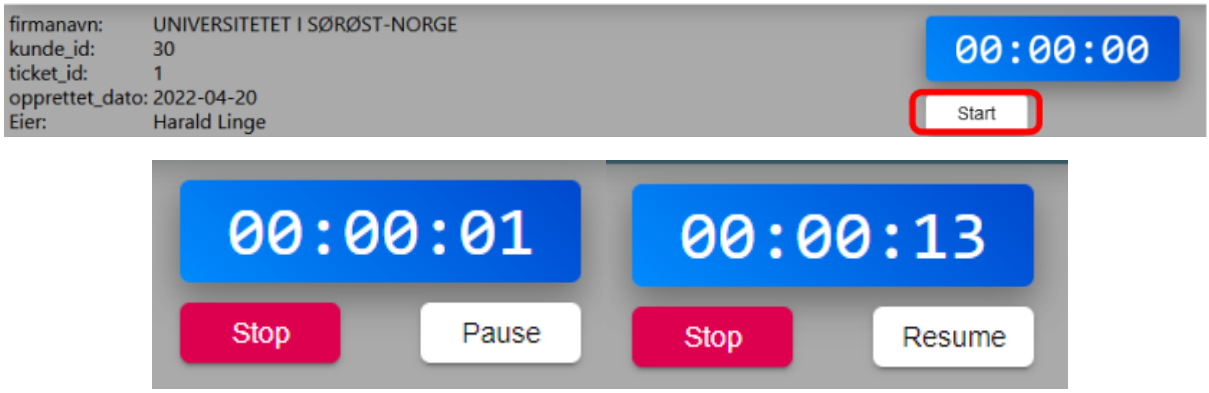
Når man starter applikasjonen vil man som oftest velge seg et kundeforhold man skal bearbeide eller jobbe på. Via "search"-feltet i ticket-panelet kan man søke opp den kunden man vil.

Med eksempel fra prototypen kan man se at man vil i tillegg få muligheten til å velge hvilken parameter man skal søke på, som for eksempel med "kundenavn" eller "organisasjonsnummer".

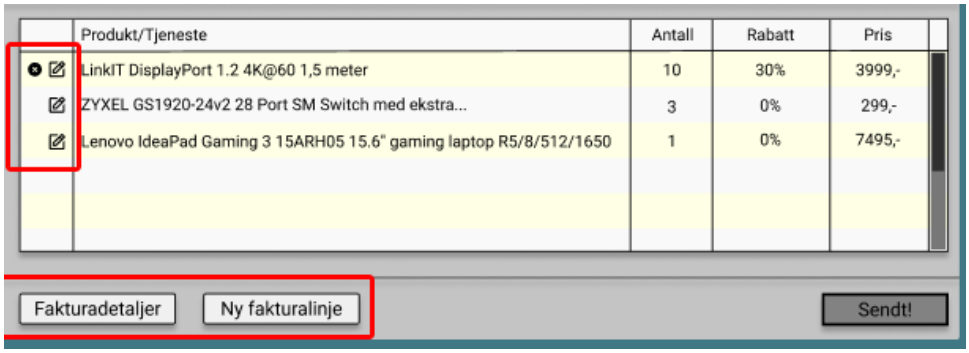
Etter man har søkt på kunde og funnet frem til riktig ticket, kan man gå videre ved å trykke på raden som representerer den. All funksjonalitet på siden vil da gjelde den valgte ticketen eller kundeforhold.

Fakturabehandling

Etter man har valgt en ticket, vil informasjon om faktura vises i vinduet under listen over kunder. Her kan man sette på en tidtaker som holder styr på hvor lenge man har jobbet med denne ticketen. Om man vil sette den på pause, trykker man på "pause", og "resume" for å fortsette. Når man er ferdig med ticketen, trykker man på "stop", og tiden registreres på den valgte ticket.

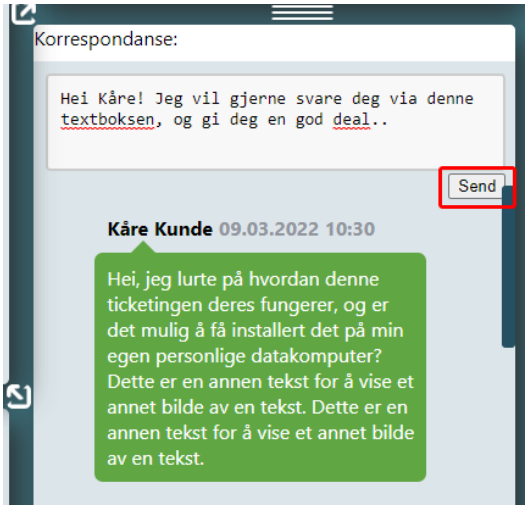


I dette fakturapanelet vil man få muligheten til å redigere fakturalinjer og legge til og fjerne artikler fra fakturaen ved å manipulere fakturaen direkte, og sende den til Fiken for fakturering. Her representert ved bilde av prototypen. Man åpner opp for å redigere en fakturalinje ved å trykke på det firkantede ikonet. Og sletter en fakturalinje ved å trykke på den sirkelen med et kryss. Ved å trykke på en av knappene under vil man enten se utfyllende detaljer om fakturaen eller legge til en fakturalinje med produkt.

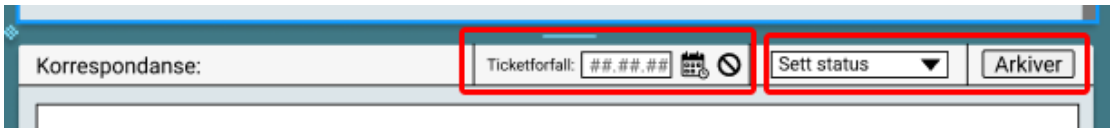


Korrespondanse

Under korrespondanse-panelet kan man se mailkorrespondansen mellom en kunde og en ansatt via den valgte ticketen. Her kan man skrive i tekstboksen som vist på bildet og trykke "send" for å sende e-posten til den valgte kunden.



Som representert med bilde av prototypen, kan man se at man vil få muligheten til å sette eller fjerne "forfall" på en ticket med en datovelger. I tillegg kan man endre status på ticketen og arkivere den om den er ferdig.

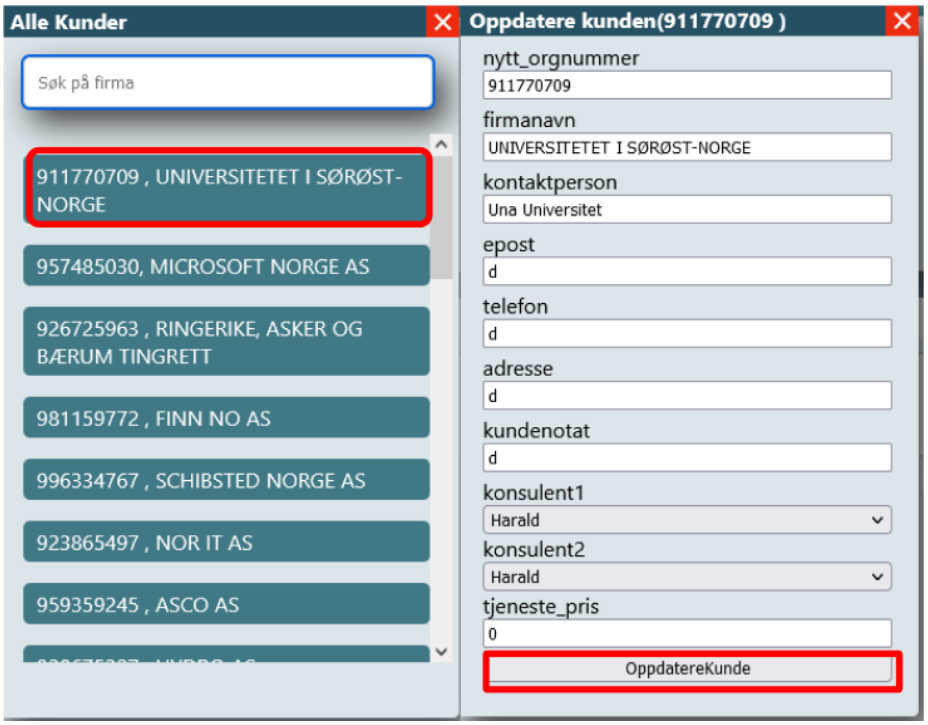


Notater

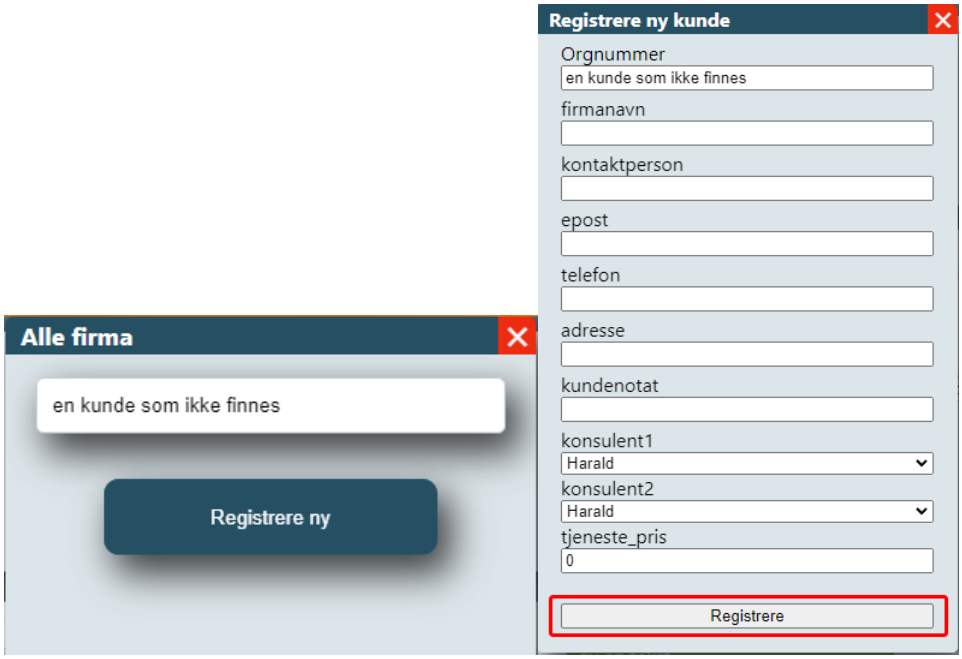
I panelet for notater kan man legge til et nytt notat med via dokument-ikonet i øvre høyre hjørne. Om man vil slette et notat kan man trykke på søppelkasse-ikonet på det gjeldende notatet og om man vil redigere det trykker man på ikonet til høyre for søppelkassen.

Kundebehandling

Om man skal oppdatere eller registrere en ny kunde, trykker man på "kundebehandling" oppe til høyre i applikasjonen åpnes et panel hvor man kan søke på eksisterende kunder og redigere de.

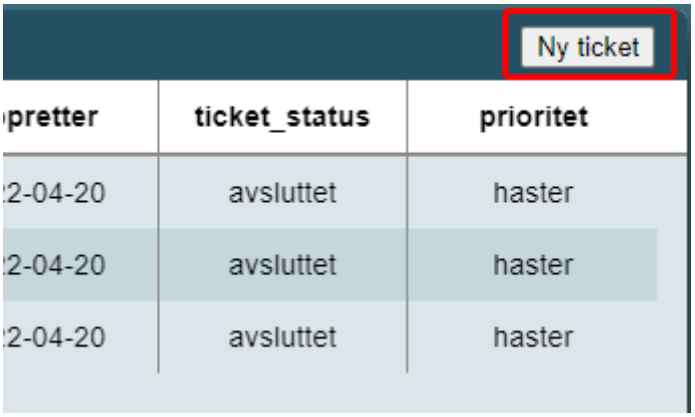


Om kunden ikke finnes, kan man legge til firmaet som en ny kunde ved å trykke på "registrere ny" og fylle inn informasjon i skjemaet som presenteres. Man lagrer ved å trykke på "registrer" på bunn av skjemaet.



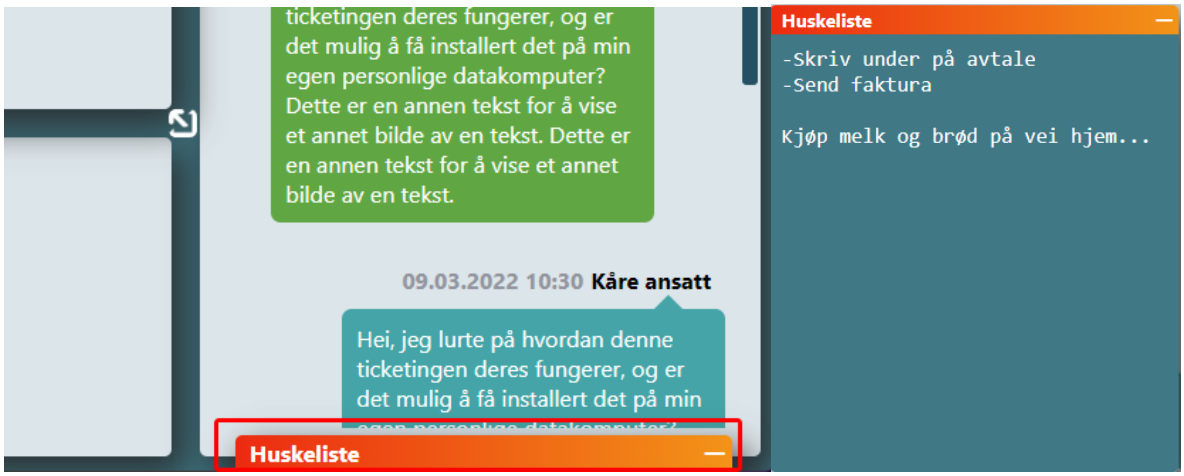
Ny ticket

Om man vil lage en ny ticket, trykker man på “Ny ticket”-knappen i ticket-panelet. Man vil bli presentert med et vindu hvor man kan skrive ned detaljer om henvendelsen, men denne funksjonaliteten er dessverre ikke ferdig enda.



Huskeliste

Som et ekstra praktisk hjelpemiddel er det en huskeliste nede til høyre av applikasjonen som man kan få tilgang til ved å dra det oransje vinduet opp. Slik man ville flyttet et vindu i Windows. For å minimere igjen trykker man på ikonet oppe i høyre hjørne av huskeliste-vinduet.



20.6 User stories

Under kan du lese ulike user story’s som vi også vil benytte når systemet skal testes. En av disse vil også bli skrevet om til en fully dressed use case. Begrunnelsen for å skrive user storys er at man skal kunne vise hvordan vi ser for oss at sluttbrukeren benytter seg av systemet og hvordan det vil være av verdi for brukeren.

Alfred Ansatt er ute på oppdrag hos kunde. #US1

Alfred Ansatt ankommer kunden og logger seg inn på sin mobil, går inn på den aktuelle kunden og starter tidtakeren. Når Alfred ansatt er ferdig hos kunden, går han igjen inn på mobilen og stopper tidtakeren før han reiser.

Alfred Ansatt sender melding til kunde #US2

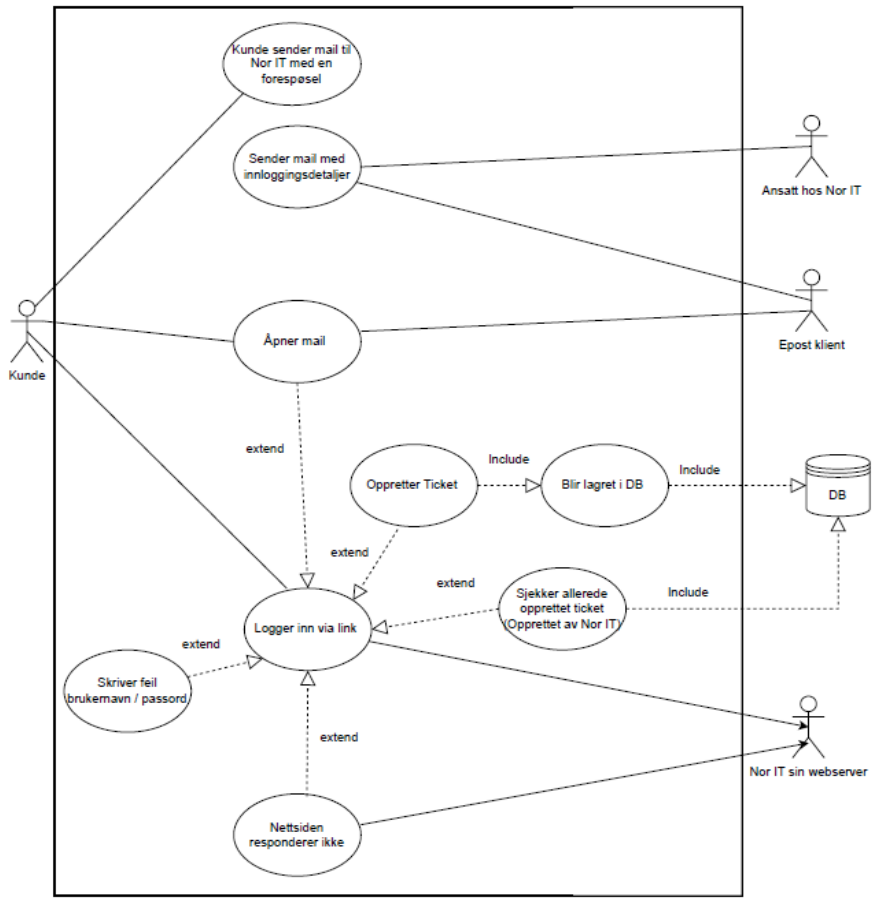
Alfred Ansatt logger seg inn for å oppdatere ticket nummer # for å gi beskjed om at bestilt vare er ankommet lageret og blir levert i løpet av morgendagen. Deretter logger han seg ut.

Kristin Konsulent oppretter kundeforhold. #US3

Kristin Konsulent åpner en melding med forespørsel om å opprette kundeforhold. Hun oppretter dette i de interne systemene og i Fiken.

20.7 Use Case diagram

Figuren under er et use case diagram fra tidlig i utviklingen vår, dette er ikke noe som er relevant for slik systemet til slutt ble, men viser hvordan vi så for oss at det skulle fungere tidligere i prosessen. Dette ble som nevnt før i rapporten revidert etter et møte med oppdragsgiver.



20.8 Use Cases

Under er flere use cases som ble laget tidlig i analysefasen. Flere av de er utgåtte med tanke på forandringer som har blirt gjort i etterkant.

En ansatt hos Nor IT fakturerer en kunde. #UC2

1 Kort beskrivelse av Use case

1.1 En ansatt hos Nor IT skal opprette en faktura som skal sendes til Fiken. Timer, tjenester og eventuelle produkter registreres på en ticket, hvorpå systemet genererer ordrelinjer og en ordre kladd. *Prisene vil være tilknyttet kunde.* Linjene blir sendt til FikenAPI med en referanse til ordrekladden. Hvor igjen en ansatt hos Nor IT logger seg inn i Fiken og enten godkjenner eller endrer på fakturakladden før den blir videresendt til kunde.

2 Aktører

2.1 Ansatte i Nor IT

2.2 Fiken API

2.3 Kunde av Nor IT

2.4 PSA systemet

3 Forhåndskrav

3.1 Nor IT har kunder som benytter seg av support, kjøper tjenester eller produkter av selskapet

3.2 Fiken-API fungerer som det skal

3.3 PSA-applikasjonen fungerer som det skal

4 Grunnleggendeflyt av hendelser

4.1 En ansatt ved Nor IT oppretter en ticket via PSA systemet

4.2 Arbeidstimer, tjenester og produkter registreres på ticketen

4.3 En ansatt ved Nor IT fakturerer ticketen via Fiken-API

4.4 Kunden mottar fakturaen

4.5 Kunden betaler for vare eller tjenesten(e)

4.6 Kvitteringen sendes gjennom fiken-API tilbake til Nor IT

5 Alternativflyt av hendelser

5.1 Fiken er nede

5.1.1 Ansatte / PSA systemet får ikke kontakt med Fiken-API

5.1.2 Den ansatte må prøve igjen ved en senere anledning

5.1.3 Se fra punkt 4.3

5.1.4 Alternativ flyt avsluttes

Alternativflyt 5.2

5.2 Kunde betaler ikke faktura

5.3 En ansatt ved Nord IT oppretter en ticket via PSA systemet

5.4 Arbeidstimer og produkter registreres på ticketen

5.5 En ansatt ved Nor IT fakturerer ticketen via Fiken-API

5.6 Kunden mottar fakturaen

5.7 Kunden betaler ikke for vare eller tjenesten(e)

5.8 Nor IT sender purring

6 Nøkkelscenarier

6.1 Fiken må være operativ og fungere som normalt

6.2 Den ansatte må ha lagt inn rett antall timer, tjenester og produkter slik at fakturaen er korrekt.

En ansatt hos Nor IT anvender huskelisten #UC3

1 Kort beskrivelse av Use case

1.1 En ansatt hos Nor IT skal benytte seg av huskelisten på webapplikasjonen.

2 Aktører

2.1 Ansatt i Nor IT

2.2 PSA webapplikasjonen

3 Forhåndskrav

3.1 Ansatt på Nor IT bruker webapplikasjonen på sin stasjonære PC

3.2 Webapplikasjonen fungerer som den skal

4 Grunnleggendeflyt av hendelser

4.1 Ansatt hos Nor IT beveger musen ned til fanen «Huskeliste»

4.2 Åpner «huskeliste» ved bruk av venstretrykk på musen

4.3 Huskeliste popper opp som et vindu

4.4 Ansatt noterer ned ting som de ved en senere anledning kan gå tilbake å se på

4.5 Lukker vinduet ved å trykke med musen oppi hjørnet på «X» i huskeliste vinduet

4.6 Huskeliste minimeres

4.7 Fanen legger seg nederst på samme sted

5 Alternativflyt av hendelser

5.1 Webapplikasjonen fungerer ikke som den skal

5.1.1 Ansatt får ikke kontakt med webapplikasjonen

6 Nøkkelscenarier

6.1 Webapplikasjonen må være operativ og fungere som normalt

6.2 Den ansatte må ha bruker slik at han/hun får tilgang til webapplikasjonen

Eksisterende kunde logger inn for første gang. #UC4

1 Kort beskrivelse av Use Case:

En eksisterende kunde for Nor IT ønsker å logge seg inn på sin side i deres portal for å sjekke at deres nye oppkobling er i orden. Dette er en kunde som logger seg inn på siden for første gang etter at de har blitt «godkjent» som kunde.

2 Aktørene

2.1 Eksisterende kunde

2.2 Ansatt hos Nor IT

2.3 Nor IT sin nettside for tickets

2.4 Nor IT sin webserver

2.5 Epost klient

2.6 Database

3 Forhåndskrav

3.1 Kunde sender mail til Nor IT og etterspør informasjon om for eks. et nytt produkt

3.2 Nor IT har sendt korrekt informasjon til kunden slik at de har alt de trenger for å logge inn

3.3 Nettsiden til Nor IT er operativ

4 Grunnleggende flyt av hendelser

4.1 Kunden følger fremgangsmåten de har fått oppgitt fra Nor IT i mail

4.2 Nettsiden responderer som normalt

4.3 Kunde følger link i mottatt mail fra Nor IT sin ansatt

4.4 Kunde skriver inn oppgitt brukernavn og passord

4.5 Kunden trykker på «logg inn» knappen

4.6 Kunden blir logget suksessfullt inn på siden

4.7 Komplett use case avsluttes

5 Alternativ flyt av hendelser

5.1 Kunden har fått oppgitt feil brukernavn eller passord

5.1.1 Kunden må ta kontakt med Nor IT via mail for å få korrekt påloggingsinformasjon

Alternativflyt 5.2:

5.2 Kunden skriver inn feil brukernavn eller passord

5.2.1 Kunden får beskjed om feil brukernavn eller passord

5.2.2 Kunden prøver på nytt og skriver nå korrekt brukernavn og passord

5.2.3 Kunden blir logget suksessfullt inn på siden

5.2.4 Alternativ flyt avsluttes

Alternativflyt 5.3:

5.3 Nettsiden til Nor IT er nede pga. feil

5.3.1 Kunden må prøve på nytt ved en senere anledning (Se fra punkt 4)

5.3.2 Use casen stopper opp uten suksess

Alternativflyt 5.4:

5.4 Nettsiden til Nor IT er nede pga. vedlikehold

5.4.1 Kunden må prøve på nytt ved en senere anledning (Se fra punkt 4)

5.4.2 Use casen stopper opp uten suksess

Alternativflyt 5.5:

5.5 Kunde oppretter ticket

5.5.1 Ticket blir opprettet

5.5.2 Alternativ flyt avsluttes

Alternativflyt 5.6:

5.6 Kunde leser ticket opprettet av ansatt hos Nor IT

5.6.1 Ticketen blir besvart av kunde

5.6.2 Alternativ flyt avsluttes

6 Nøkkel scenarier

6.1 Kunde har ikke fått korrekte opplysninger fra Nor IT på mailen ang innlogging

6.2 Nettsiden til Nor IT er nede pga. feil

6.3 Nettsiden til Nor IT er nede pga. vedlikehold

En kunde legger inn en ordre hos Nor IT, med hjelp av en konsulent #UC5

1 Kort beskrivelse av Use Case:

En eksisterende kunde av Nor IT ønsker å innhente informasjon, pris og leveringstid ang. nye produkter de ønsker og gå til anskaffelse av. Use casen vil vise hvordan kunden går frem og hvordan Nor IT besvarer henvendelsen og legger inn i en ordre for kunden.

2 Aktørene

2.1 Kunde av Nor IT

2.1 Nettsiden til Nor IT

2.3 Ansatt hos Nor IT

2.4 Systemet til Nor IT

3 Forhåndskrav

3.1 Kunde har innlogging til sin side på Nor IT for å opprette en ny ticket ang. pristilbud

3.2 Nettsiden til Nor IT er operativ

4 Grunnleggende flyt av hendelser

4.1 Kunden logger seg inn på sin side på Nor IT

4.2 Kunden oppretter en ny ticket fra menyvalget

4.3 Kunde legger inn en beskrivelse ang. hva de ønsker pristilbud på og forespørsel om leveringstid og lignende.

4.4 En konsulent hos Nor IT ser forespørselen

4.5 Konsulenten sjekker priser, produkter og leveringstider

4.6 Konsulenten besvarer ticketen til kunden med pristilbud og leveringstid

4.7 Kunden bekrefter kjøpet

4.8 Konsulenten legger inn bestillingen på ticketen, med korrekt antall og eventuelle tjenester.

4.9 Fakturalinjene blir lagt til i fakturakladd av systemet til Nor IT

4.10 Use casen avsluttes med suksess

5 Alternativ flyt av hendelser

5.1 Kunden sin forespørsel blir oversett eller glemt av konsulentene som er på jobb

5.1.1 Konsulenten oppdager forglemmelsen og svarer kunden påfølgende arbeidsdag

5.1.2 Se fra punkt 4.5

5.1.3 Alternativ flyt avsluttes

Alternativflyt 5.2:

5.2 Nettsiden til Nor IT er nede pga. feil, kunden får dermed ikke laget ticket

5.2.1 Kunden velger og ta kontakt med Nor IT på mail

5.2.2 Konsulenten som får mailen oppretter en ticket for kunden og videre kommunikasjon foregår igjennom ticketing-kanalen

5.2.3 Se fra punkt 4.5

5.2.4 Alternativ flyt avsluttes

Alternativflyt 5.3:

5.3 Nettsiden til Nor IT er nede pga. feil, kunden tar dermed kontakt med et annet firma

5.3.1 Alternativt firma løser innkjøp for kunde

5.3.1 Alternativ flyt avsluttes

Alternativflyt 5.4:

5.4 Kunde har glemt brukernavn eller passord for å logge seg inn på sin side

5.4.1 Kunde velger glemt brukernavn eller passord

5.4.2 Kunde får dette tilsendt på mail automatisk

5.4.3 Kunde logger seg inn på sin side. Se punkt 4.1

5.4.3 Alternativ flyt avsluttes

6 Nøkkel scenarier

6.1 Kunde har korrekt brukernavn og passord

6.2 Nettsiden til Nor IT er nede pga. feil

6.3 Konsulentene på Nor IT overser tickets

Kunde av Nor IT oppretter en ticket. #UC6

1. **Kort beskrivelse av UseCare Nor IT Ticketing**

   1.1 Aktøren (Nor IT) ønsker at kundene skal kunne opprette en ticket eller en case hvis de skulle lure på noe angående arbeidsprosess, eller har spørsmål angående bestillinger av service o.l.

2. **Aktører**

   2.1 Kunder hos Nor IT

   2.2 Bedriften Nor IT

   2.3 Web applikasjon (PSA)

3. **Forhåndskrav**
   3.1 PSA web applikasjonen er operativ og fungerer som den skal
   3.2 Kunden har internettilkobling
   3.3 Ansatte hos Nor IT er på jobb
4. **Grunnleggende flyt av hendelser**
   4.1 Kunden trenger mer informasjon fra Nor IT
   4.2 Kunden åpner en ticket gjennom ticketing systemet innlemmet i web applikasjonen
   4.3 Kunden sender ticketen
   4.4 En ansatt fra Nor IT åpner og besvarer ticketen, deretter sendes den tilbake til kunden
   4.5 Kunden har nå en ticket som har statusen, «venter på svar»
   4.6 Kunden åpner ticketen og leser besvarelsen fra den ansatte ved Nor IT
   4.7 Kunden føler hen fikk svar og takker for service
   4.8 En ansatt lukker ticketen og markerer den som løst.
5. **Alternativ flyt av hendelser**
   5.1 Nettside er nede eller undervedlikehold
      5.1.1 Kunden får ikke kontakt med webserver

   Alternativflyt 5.2:

   5.2 Ticketen kan bli glemt eller oversett
      5.2.1 Kunden går inn på eksisterende eller lager ny ticket og prøver å opprette kontakt med Nor IT
6. **Nøkkel scenarier**
   6.1 Ticketen ble sendt etter sistemann fra Nor IT hadde dratt hjem
   6.2 Nettsiden er nede eller drives vedlikehold på

Hvordan behandles Tickets av de ansatte hos Nor IT. #UC7

1. **Kort beskrivelse av Use Case Nor IT Ticketing**
   1.1 Hvordan behandles Tickets av de ansatte hos Nor IT
2. **Aktører**
   2.1 Kunder hos Nor IT
   2.2 Ansatte hos Nor IT

2.3 Web applikasjon (PSA)

3. **Forhåndskrav**
   3.1 PSA web applikasjonen er operativ og fungerer som den skal
   3.2 Ansatte hos Nor IT er på jobb
   3.3 Ansatte hos Nor IT er kjent med ticketing systemet
   3.4 Nor IT har Tickets som krever behandling
4. **Grunnleggende flyt av hendelser**
   4.1 En ansatt fra Nor IT finner en ubesvart Ticket
   4.2 En ansatt åpner en ticket gjennom ticketing systemet innlemmet i web applikasjonen
   4.3 En ansatt svarer på ticketen og sender den tilbake til kunden (SMTP)
   4.4 Kunden har nå en ticket som har statusen, «venter på svar»
   4.5 Kunden åpner ticketen og leser besvarelsen fra den ansatte ved Nor IT
   4.6 Kunden føler hen fikk svar og takker for service
   4.7 En ansatt lukker ticketen og markerer den som løst. (kanskje kunden skal ha mulighet til dette med tanke på at han vet best om problemet er løst eller ikke – spør Olav?)
5. **Alternativ flyt av hendelser**
   5.1 Nettside er nede eller undervedlikehold
      5.1.1 De ansatte får ikke tilgang til ticketene

   Alternativflyt 5.2:

   5.2 Ticketen kan bli glemt eller oversett
      5.2.1 De ansatte glemte å besvare ticketen eller så den ikke i stacken med tickets.
      5.2.2 En ansatt lukket en ticket ved uhell før den var løst
6. **Nøkkel scenarier**
   6.1 Ticketen ble sendt etter sistemann fra Nor IT hadde dratt hjem
   6.2 Nettsiden er nede eller drives vedlikehold på

20.9 Møtereferater

For hvert av møtene vi har hatt, har det vært en agenda og vi har tatt notater fra møtene som vi har hatt med oppdragsgiver Olav hos Nor IT. Dette er gjort slik at vi skal ha noe å se tilbake på, om vi skulle bli usikre om hva som har blitt tatt opp under de ulike møtene, og

ikke minst for å kunne huske på alle svarene vi har fått på spørsmålene våre. Fra 17.6 kan dere lese utdrag av møtenotatene. I utdraget har vi fokusert på det som har vært hovedtema og som dermed har vært det viktigste for møtene for de ulike dagene. Det vil også ligge notater her fra fellesmøtene vi skal ha med oppdragsgiver Olav og veileder Rania El-Gazzar.

Første møte med Olav Gulbrandsen fra Nor IT
Dato: 18.01.2022 Kl. 11:00-13:00

Til stede: Henrik, Halvor, Kristian, Ketil, Elin og Melli. *Oppdragsgiver* Olav fra Nor IT.

Lokasjon: Kontoret til Nor IT på Follum, og noen på teams.

Agenda: Bachelor skissen, Timeregistrering, Ticket og case, Fakturering, vi ønsker også å se på noen faste dager for møte med Olav, kan selvsagt endres på underveis ved behov.

Notater:

Bachelorskissen ser bra ut og vi har fått med mer eller mindre alt Olav har sett for seg.

Ticketing og caser blir benyttet om hverandre. Tickets skal kun være for eksisterende kunder. Prospekter / nye kunder må ta kontakt på telefon eller mail, først når man ser det blir et kundeforhold vil det bli opprettet logg inn funksjon for kunden og dermed også tilgang til tickets.

Fakturering bør skje ca. hver 14-dag. Kan også gjøres fortløpende om det kun er et produkt kunden ønsker. I dag skjer faktureringen manuelt, hvor det blir benyttet Excel ark. (Vi vil få et eksempel tilsendt)

Møter vil bli avholdt hver 14 dag i første omgang, men dette kan endres undervis. Vi ønsker å ha møter når vi har noe å vise til eller deler som er ferdige, som vi ønsker tilbakemelding på. Men også for å kunne samle opp å spørre spørsmål underveis.

Andre møte med Olav Gulbrandsen fra Nor IT
Dato: 01.02.2022 Kl. 14:00-15:30

Til stede: Henrik, Halvor, Kristian, Ketil, Elin og Melli. *Oppdragsgiver* Olav fra Nor IT.

Lokasjon: Kontoret til Nor IT på Follum.

Agenda: Mer om tickets, noen spørsmål ang Azure, foreslå dato for møter sammen med Rania, ønsker også å vite litt mer ang Fiken

Notater: Kan være behov for å ha en stoppeklokke funksjon på tickets, for å kunne starte tidtaking, men også kunne pause den om man jobber mer flere samtidig. Tiden må også kunne endres i ettertid ved behov. Ønsker mulighet for å legge inn «private» notater på tickets, som ikke kundene kan se.

Olav sjekker opp mulighetene for subscription hos Azure og kommer tilbake til oss ila uken.

Vi sender e-post og hører med Rania ang møtedato, kommet med forslag om møte den 09.03.2022 og den 06.04.2022.

Når det kommer til Fiken kan vi få noen screenshots av Olav, eventuelt være med en dag og se når de kjører noen fakturaer. Ikke noe som haster enda.

Tredje møte med Olav Gulbrandsen fra Nor IT
Dato: 16.02.2022 Kl. 14:00-15:30

Til stede: Henrik, Halvor, Kristian, Ketil, Elin og Melli. *Oppdragsgiver* Olav fra Nor IT.

Lokasjon: Kontoret til Nor IT på Follum.

Agenda: Hvor mange bankkontoer har Nor IT, lagerbeholdning, vise frem wireframes, snakke om innlogging for kundene

Notater: Når det kommer til kontoer har de kun en konto det kommer fakturapenger inn på. Videre har de kun et lite lager med noen småting her og der, så det er ikke noe vi trenger å ta hensyn til. Wireframes for tickets, må ha med en «lovet dato» slik at de tar kontakt med kunden når de har sagt det. Viktig at dette kommer tydelig frem. Tickets som er lukket trenger ikke å være synlig, må kunne hentes opp selvsagt. Kan være ønskelig med en søkefunksjon på tickets, men dette er «nice to have». Når det kommer til Fiken vil det nok være begrensninger på antall tegn, så det må komme tydelig frem hva som har blitt fakturert, men med få ord.

Tickets må også inneholde fakturerbar / ikke fakturerbar tid, mulig i kombinasjon med en nedtrekks liste for rådgivning, teknisk bistand og reisetid, som faktureres halv pris. Det finnes også kunder som har forhandlet seg frem til spesialpris, her kan vi koble opp timepris mot

kunde for eks. Ellers synes Olav wirefram'ene ser fine ut, og vil ikke legge for mye føringer for oss.

Når det kommer til innlogging for kundene mener oppdragsgiver at dette er noe som kan være for fase 2, altså ikke en del av vårt scope. Vi sier oss enige i at det kan bli tidkrevende, og vil se om vi kan få til en løsning hvor kommunikasjon kan gå igjennom en ticket uten at kunden må logge inn. Dvs. om vi kan få til en mail korrespondanse eller lignende.

Første møte med veileder Rania El-Gazzar & Olav Gulbrandsen fra Nor IT
Dato: 09.03.2022 Kl. 12:00 – 13:30

Til stede: Henrik, Halvor, Kristian, Ketil, Elin og Melli. *Oppdragsgiver* Olav fra Nor IT. *Veileder* Rania El-Gazzar.

Lokasjon: Møterom på USN campus Ringerike

Agenda: Tilbakemeldinger på prototypen. Funksjonalitet på mobil VS desktop. Spørsmål til Rania rund levering av kode, og hva hun tenker tidsmessig om det som gjenstår. Eventuelt andre gode innspill fra Rania

Notater: Fra Olav får vi gode tilbakemeldinger på prototypen, han kan ikke se noe som mangler umiddelbart og mener vi har fått på plass ønskene han har kommet med fra tidligere møter. Siden dette er et system som vil bli litt smått og bruke på mobil, så vektlegges det at når det kommer til funksjonalitet på mobil, er egentlig det aller viktigste at vi får implementert muligheten for at de ansatte kan registrere tiden de bruker når de er ute hos kundene. Når det kommer til logg inn muligheter for kunder, så vil dette være noe som tilfaller om vi får tid til det. Vi får gode tips og tanker fra både Rania og Olav når det kommer til videre utarbeidelse av både rapport og selve systemet.

Andre møte med veileder Rania El-Gazzar & Olav Gulbrandsen fra Nor IT
Dato: 22.04.2022 Kl. 12:00 – 14:00

Til stede: Henrik, Halvor, Kristian, Ketil, Elin og Melli. *Oppdragsgiver* Olav fra Nor IT. *Veileder* Rania El-Gazzar.

Lokasjon: Møterom på USN campus Ringerike

Agenda: Frontend, Api, devtools, databasemodell. Veien videre med generelle spørsmål spesielt til Rania ang rapport og videre utførelse av denne.

Notater: Gode tilbakemeldinger på arbeidet vi viser frem og gode råd for veien fremover. Viktig og holde motivasjonen oppe og reflektere rundt valgene vi har tatt. Ikke starte med noen nye ting nå, men fullføre det vi har startet på, slik at vi ikke mister tid. Når det kommer til rapporten vil det legges mest vekt på prosessen, men vi må huske å legg ved en god brukerveiledning av produktet. Passe på endelig bruker og tar hensyn til at de har forståelse for hvordan man bruker systemet. Metodikken må vi tenke godt igjennom! Må også ha med et avsnitt ang sikkerhet og hvilke tanker vi har gjort oss rundt dette.

Møte med veileder Rania El-Gazzar
Dato: 22.04.2022 Kl. 12:00 – 14:00

Til stede: Henrik, Halvor, Kristian, Ketil, Elin og Melli. *Veileder* Rania El-Gazzar.

Lokasjon: Møterom på USN campus Ringerike

Agenda: Tilbakemelding på rapport og gjennomgang av spørsmål vi har med til møtet.

Notater: Rania går først igjennom sine tanker rundt rapporten, ang hvilke deler av rapporten som må omstruktureres, med tanke på hva som skal flyttes til vedlegg og hvordan figurer må nummereres blant annet. Vi har også med noen spørsmål til Rania som vi får en gode svar på.