Norges Handelshøyskole
Bergen, Vår 2022

# Få fart på regnskapet!

*En designvitenskapelig studie som undersøker hvorvidt lavkodeverktøy egner seg til effektivisering av arbeidsprosesser i regnskapsførerselskap.*

**Marie Grung og Charlotte Heggland Søvik**

**Veileder: Kjersti Berg Danilova**

Masterutredning i økonomi og administrasjon

Hovedprofil: Business Analytics

NORGES HANDELSHØYSKOLE

# Forord

Denne masteroppgaven er skrevet som en del av masterutdanningen i økonomi og administrasjon ved Norges Handelshøyskole, innenfor hovedprofilen Business Analytics.

Vi ønsker å rette en stor takk til alle som har bidratt til denne masteroppgaven. Takk til vår veileder Kjersti Berg Danilova for din tid, oppfølging og konstruktive tilbakemeldinger. Det hadde ikke vært mulig å skrive denne masteroppgaven uten medvirkning fra næringslivet. Takk til Avo Consulting AS for gode råd og for å ha satt oss i kontakt med Appfarm AS. Teknologien til Appfarm har vært avgjørende for at vi kunne gjennomføre vår studie. Vi er takknemlig for at vi har fått muligheten til å bruke deres tjenester og vil takke selskapet og kontaktpersonene som har hjulpet oss med tilganger, informasjon og en evaluering av utviklingsarbeidet. Til slutt vil vi rette en takk til Økonomihuset AS for gode samtaler som ga innsikt i deres arbeidshverdag og tilgang til demomiljøet i Poweroffice GO.

Norges Handelshøyskole

Bergen, juni 2022

Marie Grung og Charlotte Heggland Søvik

# Sammendrag

Digitalisering i regnskapsbransjen gjør at konkurransesituasjonen, arbeidsprosesser og kompetansebehov er i endring. Forskning har i liten grad fokus på hvilken teknologi regnskapsførerselskapene bør benytte i den digitale omstillingen. Formålet med studien er å undersøke hvordan lavkodeteknologi kan brukes til digitalisering i et regnskapsførerselskap. Norge har behov for flere programvare- og applikasjonsutviklere. Lavkode fremstilles som et potensielt bidrag for å løse dette behovet da det setter flere i stand til å delta i applikasjonsutvikling gjennom visuelle og brukervennlige grensesnitt.

I denne masteroppgaven anvender vi designvitenskpelig forskningsmetode. Metoden innebærer nyskapning gjennom design av nye eller innovative artefakter. Vi har benyttet en lavkodeplattform til å utvikle en applikasjon som automatiserer en arbeidsprosess for et regnskapsførerselskap. Studien gir dermed et konkret eksempel på hvordan lavkodeteknologi kan benyttes og anvendes i praksis.

Gjennom arbeidet erfarer vi at lavkodeplattformen er både intuitiv og fleksibel. Vi lykkes i å utvikle en fungerende applikasjon på kort tid og konkluderer med at lavkodeverktøy kan benyttes av personer uten særlige forkunnskaper. Den ferdige løsningen er integrert med et regnskapsystem og eliminerer behovet for manuelle oppgaver i den aktuelle arbeidsprosessen. Dette reduserer risiko for menneskelige feil og sikrer automatisert verdiskapning i regnskapsførerselskapet. Vi konkluderer med at lavkodeverktøy kan bidra til den digitale utviklingen i regnskapsførerselskap.

***Nøkkelord*** – NHH, masteroppgave, lavkode, regnskap.

# Innhold

# Figurliste

# Tabelliste

# 1 Innledning

Digitalisering påvirker i dag de fleste delene av samfunnet. I Næringslivets Hovedorganisasjons (NHO) siste kvartalsrapport for 2021 kommer det frem at Norge er et av de mest digitaliserte landene i verden (Næringslivets Hovedorganisasjon, 2021). Når det gjelder bruk av IT i bedrifter ligger likevel Norge i snitt 19 prosent bak det ledende landet. Dette kan tyde på at det finnes et potensiale for ytterligere digitalisering i bedriftene.

Regnskapsbransjen regnes som en av de mest utsatte bransjene når det kommer til digitalisering og automatisering av arbeidsoppgaver (se for eksempel Frey og Osborne, 2013). Historisk sett har arbeidet til regnskapsførere utviklet seg i takt med teknologien (Granlund og Mouritsen, 2003). Bransjen har hatt to store skift, først ved overgangen fra manuelle systemer til funksjonelle IT-systemer og deretter introduksjonen av mer komplekse og sammensatte systemer (Heggernes, 2020). Bransjen opplever nå et tredje skift, omtalt som digitalisering (Karimi og Walter, 2015; Knudsen, 2020; Parviainen et al., 2017). Dette medfører endrede konkurransebetingelser, markedsbehov og forventninger til regnskapsførerselskapene (Larsen, 2021).

Digitalisering bidrar til å fjerne repeterende oppgaver som har lav verdi (Knudsen, 2020) og det estimeres at bedrifters avhengighet av regnskapsførere reduseres i takt med utviklingen (Appelbaum og Nehmer, 2017). Det stilles spørsmål ved hva regnskapsførere skal gjøre når store deler av regnskapet er automatisert (for eksempel Marrone og Hazelton, 2019). Både forskere og bransjeorganisasjonen Regnskap Norge er enige om at regnskapsførerselskapene bør ha større fokus på rådgivning i fremtiden (Schei et al., 2015; Kvalheim, 2020). For å dra nytte av utviklingen må regnskapsførerselskapene tilpasse sin kompetanse til teknologien som brukes (Marrone og Hazelton, 2019).

Plattformeiere har mye makt i dagens digitale samfunn (Kornberger et al., 2017). Dette gjelder også i regnskapsbransjen hvor det pekes på høye og uoversiktlige kostnader (Kvalheim, 2022). I tillegg er markedet dominert av få, store leverandører (Austheim, 2020). Til tross for at mange av arbeidsoppgavene i regnskapsførerselskap er automatisert ved bruk av moderne regnskapssystemer, innebærer fortsatt regnskapsarbeid en del manuelle prosesser (Baksaas et al., 2019). I 2018 utgjorde regnskapstjenester 75 prosent av omsetningen i bransjen (Regnskap Norge, 2019). Noen av de manuelle prosessene

kunne trolig vært automatisert ved bruk av tilpasset eller spesialutviklet programvare. Tradisjonelt sett krever slik tilpasning utviklere med kunnskap om blant annet kodespråk, men det er ikke nødvendigvis tilfellet lenger. Kanskje kan regnskapsførerselskapene selv stå for utvikling av enkle løsninger som bidrar til å effektivisere arbeidsprosesser, opprettholde konkurranseposisjon og være relevant i fremtiden?

> *"The future of coding is no coding at all"*
>
> —Chris Wanstrath, CEO at GitHub (GitHub, 2017, 52:49)

Begrepet lavkode (low-code) er relativt nytt og ble først promotert av Forester i 2014 (Richardson og Rymer, 2014). Lavkode innebærer at utviklere bruker minimalt med kodespråk for å lage en applikasjon. Gartner (2021) spår at 70 prosent av alle nye applikasjoner vil være laget med lavkode- eller nullkodeteknologi innen 2025. Applikasjonene vil bli laget av teamene som skal bruke de, fremfor spesialiserte utviklere (Gartner, 2021).

Det finnes lite forskning på hvilke teknologi regnskapsbyråene skal benytte for å være konkurransedyktige og hvilke konkrete tiltak som må gjøres for å automatisere regnskapet. Lavkodekonseptet er også lite forsket på, og det finnes ingen informasjon om lavkode i regnskapsbransjen, dette til tross for at det estimeres en sterk økning i bruken av teknologien.

Formålet med denne masteroppgaven er å undersøke hvorvidt regnskapsførerselskap kan utnytte lavkodeteknologi til å effektivisere arbeidsprosesser, skape merverdi og opprettholde konkurranseposisjonen i den digitale tidsalder. Oppgavens forskningsspørsmål er følgende:

> *"Er lavkodeverktøy egnet til å automatisere arbeidsoppgaver i regnskapsbransjen og bidra til regnskapsførerselskapenes digitale utvikling?"*

Med utgangspunkt i casebedriften Økonomihuset AS undersøker vi problemstillingen ved å benytte designvitenskapelig forskningsmetode. Som en del av metoden bruker vi et lavkodeverktøy til å utvikle en applikasjon for å automatisere en prosess i bedriften. Formålet med utviklingen er å vurdere hvorvidt lavkode er egnet i regnskapsbransjen. En praktisk og virkelighetsnær tilnærming tilrettelegger for at vårt kunnskapsbidrag kan kommuniseres til bransjen og anvendes i praksis.

## 1.1 Struktur

Denne oppgaven er delt i seks kapitler. Kapittel 2 gir en innføring i relevant forskning på digitalisering i regnskapsbransjen og lavkode som konsept. I kapittel 3 redegjør vi for anvendt forskningsmetode og rammeverk. Kapittel 4 beskriver utviklingsprosessen og resultatet av arbeidet. Deretter diskuterer vi våre funn opp mot relevant teori og eksisterende forskning. Dette danner et grunnlag for å besvare forskningsspørsmålet. Avslutningsvis konkluderer vi med kunnskapsbidrag og redegjør for begrensinger ved studien og forslag til videre forskning.

# 2 Litteratur

I dette kapittelet presenterer vi litteratur og bakgrunn som er relevant for å besvare oppgavens problemstilling. Først redegjør vi for sentral teori om regnskapsbransjen med fokus på digitalisering. Deretter går vi inn på informasjonssystemer og de ulike kunnskapsregimene før vi presenterer lavkodekonseptet. Formålet er å introdusere leseren for konseptets opphav og gi en oversikt over arkitekturen og funksjonaliteten til en lavkodeplattform. I tillegg beskriver vi motiv for og utfordringer ved å benytte lavkodeverktøy.

## 2.1 Digitalisering i regnskapsbransjen

Vi befinner oss på randen av den fjerde industrielle revolusjonen som fundamentalt vil endre måten vi lever, arbeider og forholder oss til hverandre på (Schwab, 2017). På tvers av alle bransjer vil teknologiene som ligger til grunn for den fjerde industrielle revolusjonen ha stor betydning for næringslivet. Den digitale tidsalder startet på midten av 1900-tallet og markerer skiftet fra mekanisk og analog teknologi til digital teknologi (Schwab, 2017). Dette er i stor grad beskrivende for begrepet digitisering som innebærer å konvertere data og komponenter fra et analogt eller fysisk format, til et digitalt format (Osmundsen et al., 2018; Yoo, 2010). Digitisering er en forutsetning og en driver for digitalisering. I en undersøkelse utført av KPMG (2020), på oppdrag fra Kommunal- og moderniseringsdepartementet, forstår hovedvekten av deltakende norske virksomheter digitalisering som effektivisering av prosesser. I denne oppgaven definerer vi digitalisering som bruken av digital teknologi til å endre på sosio-tekniske strukturer (Osmundsen et al., 2018, s. 5) og dermed skape muligheter for nye inntekter (Gartner, u.å.), forbedre kunderelasjoner og gi merverdi til hele økonomien (Reis et al., 2019).

Regnskapsprosesser kan defineres som "arbeidsflyten med å samle, bearbeide og rapportere regnskapsinformasjon" (Baksaas et al., 2019, s. 75). Regnskapsførsel som fagområde stammer fra den tiden sivilisasjoner oppstod og forandringene i bransjen har naturligvis vært store.

> *"Da jeg begynte i bransjen på 90-tallet førte vi regnskap for hånd på papir. Det var kun et par mann på kontoret som hadde tilgang til en datamaskin."*
>
> —Arild Spandow, CEO i Amesto (Spandow, 2014).

Historisk har arbeidet til regnskapsførere utviklet seg i takt med fremskritt innen IT (Granlund og Mouritsen, 2003). De manuelle, papirbaserte rutinene ble først flyttet over til elektroniske verktøy uten at selve prosessen ble vesentlig endret. Senere, nærmere bestemt på begynnelsen av 2000-tallet, kom ERP-systemer (Enterprise Resource Planning) på markedet (Heggernes, 2020). Betegnelsen ERP beskriver et bredt system som ofte inkluderer regnskap, produksjon, salg, service og kundehåndtering (Heggernes, 2020). I dag er vi inne i den tredje fasen, omtalt som digitalisering (Karimi og Walter, 2015; Parviainen et al., 2017), hvor IT-verktøy er blitt avgjørende for regnskapstjenesten (Kvalheim, 2018). Bransjen preges av nye produksjonssystemer, konkurransebetingelser og endringer i kundenes behov og forventninger (Larsen, 2021).

Som et resultat av den digitale utviklingen har systemene blitt en vesentlig kostnadsdriver for regnskapsførersleskap. Prisen på regnskapssystemer kan være høy og oppleves som lite sammenlignbar (Kvalheim, 2022). Regnskap Norge (2021a) har gjennomført en prisundersøkelse som viste at flere av aktørene er flinke til å ta seg betalt. De omtaler praksisen i bransjen som en omvendt Netflix-modell hvor systemleverandør tar seg "betalt både per kunde, per bruker, per transaksjon og kanskje også per litt-av-hvert" (Kvalheim, 2022). Plattformeiere har mye makt i dagens digitale tidsalder (Kornberger et al., 2017). Når en plattform øker brukerbasen, øker samtidig plattformeierens innflytelse. Makt fordeles på færre og færre aktører, noe som resulterer i unaturlige monopoler eller oligopoler (The Economist, 2019). Dette er også tilfellet i regnskapsbransjen i Norge, hvor 71 prosent av alle medlemsbedriftene i Regnskap Norge bruker et system eid av Visma (Austheim, 2020). Systemleverandører og digitale plattformer kan derfor være en variabel som former regnskapsbransjen og rollen til regnskapsførere (Knudsen, 2020). Arkitekturen i ERP-systemer kan gjøre det komplisert og dyrt å utvikle gode digitale løsninger. Enkelte programleverandører kan derfor forsinke den digitale utviklingen, noe som også kan være et strategisk valg fra dominerende leverandører som ønsker å utnytte et marked som gir god inntjening (Gustavsen og Baksaas, 2019).

Teknologien fjerner repeterende oppgaver og regnskapsførere må bevege seg fra passiv regnskapsførsel til aktiv rådgivning (Schei et al., 2015). Ettersom regnskapsførerselskap i hovedsak består av regnskapsførere, er den mye omtalte rollendringen avgjørende for utviklingen til selskapene. Det estimeres at den teknologiske utviklingen vil redusere bedrifters avhengighet av regnskapsførere (Appelbaum og Nehmer, 2017). Behovet for regnskapsførere, slik vi kjenner dem i dag, vil likevel ikke forsvinne over natten (Agnew, 2016), men de nødvendige ferdighetene vil sannsynligvis endre seg (Kokina og Davenport, 2017). Da kreves det også at utdanningsforløpet oppdateres tilsvarende for å sikre at nyutdannede har arbeidsrelevant kunnskap i møte med arbeidsplassen (Al-Htaybat et al., 2018). Nyutdannede bør beherske grunnleggende tekniske konsepter for å kunne optimalisere samhandlingen mellom maskin og menneske (Baksaas, 2019).

Som et resultat av digitalisering referer en rekke studier til en virkelighet der regnskapsførernes rolle utfordres av andre yrkesgrupper og hvisker bort rammene (Knudsen, 2020; Hazgui og Gendron, 2015) for selve regnskapsfaget (Agostino og Sidorova, 2017; Arnaboldi et al., 2017b; Suddaby et al., 2015). Andre fagområder, som for eksempel IT, kan bruke digitale teknologier som inngangsbillett til regnskapsbransjen (Arnaboldi et al., 2017a). Når oppgaver som tidligere ble ivaretatt av regnskapsfører overtas av andre fagpersoner med mer digital kunnskap, risikerer regnskapsførere å miste legitimitet og innflytelse (Viale et al., 2017). Denne utviklingen setter spørsmål ved hva regnskapspraksis er, og hva regnskapsførerens rolle bør være fremmover.

Det stilles spørsmål om hvorvidt den teknologiske utviklingen vil muliggjøre nye og meningsfulle arbeidsoppgaver eller om regnskapsrollen vil bli overflødig (Lent, 2018). Noen peker på hvordan digitaliseringen ikke vil endre komponentene i regnskapet, men snarere dens indre dynamikk (Knudsen, 2020). For eksempel vil ikke en digital robot fjerne oppgaver, men endre oppgavene som må håndteres av menneskelige aktører (Knudsen, 2020). Det hevdes at regnskapsførere uten interesse for systemer er en utdøende rase, og at en regnskapsfører i dag er nødt til å ha systemkompetanse for å være verdifull (Spandow, 2014). Ferdighetene til regnskapsførere må endres tilsvarende teknologiene som endrer bransjen (Marrone og Hazelton, 2019). Dette gir regnskapsførere mulighet til å ta en mer fremtredende plass i utviklingen (Marrone og Hazelton, 2019).

Vi har ikke lykkes i å finne forskning som viser hvor langt rollendringen og digitalisering i

bransjen er kommet. Noen hevder at det er mer prat enn faktisk digital endring (Riddell, 2016). I følge Regnskap Norge (2019) stammet kun 7,5 prosent av all omsetningen fra rådgivning og konsulenttjenester i 2018. Jensen og Borge-Hansen (2020) erfarer at det største hinderet for automatisering ikke er mangel på teknologi, men heller mangel på standardisering og den reelle viljen i organisasjonen til å endre seg. De mener at det vil være nødvendig for regnskapsbransjen å overkomme dette problemet for å tiltrekke seg kvalifisert arbeidskraft i fremtiden (Jensen og Borge-Hansen, 2020). Det er ikke lenger et spørsmål om hvorvidt selskapene vil tilpasse seg den digitale utviklingen, men heller hvor fort det vil skje (Delacruz og Lim, 2018).

Begrepet *digital transformasjon* forbindes gjerne med behovet for å opprettholde konkurranseevne i en digital tidsalder. *Digital innovasjon* innebærer å kombinere digitale og fysiske komponenter på nye måter for å skape nye produkter (Yoo, 2010). Sammen med digitalisering kan digital innovasjon utgjøre en transformasjon når det over tid muliggjør vesentlige endringer i måten man arbeider på (Osmundsen et al., 2018). Utviklingen i regnskapsbransjen kan deles opp i følgende fire fokusområder som driver den digitale transformasjonen (Prem, 2015):

- digital innsamling, prosessering og analyse av data
- automatisert verdiskapning
- integrasjon mellom systemer
- digitale internettgrensesnitt mot kunden

Graden av digitalisering i bransjen påvirkes i stor grad av sluttkundenes størrelse. Omfattende digitale løsninger er ofte ikke lønnsomt for de minste kundene og innad i et byrå vil det derfor være stor forskjell på hvor digitalisert det enkelte regnskapsoppdrag er (Baksaas et al., 2019).

Baksaas et al. (2019) deler digitaliseringen hos regnskapskunder i en skala med seks nivåer hvor første nivå er "helt manuell regnskapsføring" og siste er en "perfekt flyt". Kun et fåtall av kundene i studien har nådd toppen av digitaliseringsskalaen, og det er gjerne bare noen av prosessene som er kommet så langt. Noen kunder har for eksempel et fullt integrert system for behandling av reiseregninger og utlegg (nivå seks), mens varetellingen fortsatt er helt manuell (nivå én). Studien underbygger påstanden om at regnskapsarbeid fortsatt

innebærer manuelle prosesser og at det er rom for ytterligere digitalisering i bransjen.

## 2.2 Informasjonssystemer

Digitaliseringen i regnskapsbransjen baserer seg på informasjonsteknologi (IT). IT referer til alle typer datamaskinbaserte verktøy som utnyttes til å bearbeide informasjon og som støtter en organisasjons informasjon og behov for informasjonsprosessering (Rainer og Prince, 2021, s. 4). I Norge brukes både Informasjons- og kommunikasjonsteknologi (IKT) og IT for å beskrive teknologi der informasjon bearbeides, lagres og formidles i digitalt format (Rossen, 2019a). Informasjonssystemer (IS) samler, prosesserer, arkiverer, analyserer og formidler informasjon for en bestemt hensikt (Rainer og Prince, 2021, s. 4). Hensikten ved IS kan beskrives som å skaffe rett informasjon til riktige personer, til rett tid, i riktig lengde og riktig format (Rainer og Prince, 2021, s. 12).

Ettersom informasjonssystemer skal levere informasjon må det også redegjøres for nærliggende begreper. Data er en elementær beskrivelse av ting, aktiviteter, begivenheter og transaksjoner som lagres og klassifiseres, men som ikke er behandlet og organisert til å formidle et budskap (Rainer og Prince, 2021). Informasjon er data som er organisert slik at det gir mening og verdi til mottakeren (Rainer og Prince, 2021). Kunnskap er data og/eller informasjon som er prosessert til å formidle forståelse, erfaring, akkumulert læring og ekspertise (Rainer og Prince, 2021).

### 2.2.1 To kunnskapsregimer

Bygstad (2017) presenterer terminologien lettvekt-IT og tungvekt-IT for å beskrive to kunnskapsregimer i dagens teknologiske klima. Tungvekt-IT er et etablert og modent fagfelt basert på utprøvd digital teknologi (Bygstad, 2017). Kunnskapsregimet er drevet av fagpersoner innen IT og omfatter backend løsninger som for eksempel ERP- eller regnskapssystemer. Lettvekt-IT er et fremvoksende kunnskapsregime hvor personer uten IT-utdanning spiller en større rolle (Bygstad, 2017). Lettvekt-IT innebærer blant annet mobilapplikasjoner og sensorer som muliggjøres av tingenes internett og forbrukerisering av teknologi (Bygstad, 2017). Forskjellene mellom lettvekt- og tungvekt-IT kjennetegnes ved den digitale infrastrukturen og ikke teknologien i seg selv (Bygstad, 2017). Digital infrastruktur er nettverket av sammenkoblede systemer, dette inkluderer teknologi, brukere

og utviklere (Hanseth og Lyytinen, 2010).

Tungvekt-IT blir stadig mer profesjonalisert mens lettvekt-IT drives av innovasjon (Bygstad, 2017). Lettvekt-IT er gjerne billig og brukervennlig. Dette setter andre enn profesjonelle fagpersoner i stand til å eksperimentere med rimelig teknologi. Det har for eksempel lenge vært en eksplosiv vekst av tredjeparts applikasjonsutviklere som illustrer denne trenden (Bergvall-Kåreborn og Howcroft, 2011). Lettvekt-IT egner seg for oppgaver som tungvekterne ofte ikke støtter, nemlig de enkle og umiddelbare behovene til en bruker. Derfor er lettvekt-IT nyttig for å effektivisere enkle arbeidsprosesser slik som den vi undersøker i denne masteroppgaven. Til tross for at de to kunnskapsregimene beveger seg i ulik retning er de komplementære og gjensidig avhengig av hverandre. Tungvekt-IT kan fungere som en plattform for innovativ lettvekt-IT. Lettvekt-IT tilbyr en arena for innovasjon som er utenfor omfanget av tungvekt-IT. Et fremtredende eksempel på kombinasjoner av tungvekt- og lettvekt-IT er plattformsystemer. Lavkodeverktøy som beskrives og diskuteres ytterligere i kapittel 2.3, tilbys via plattformsystemer og er godt egnet for utviklingen som karakteriserer lettvekt-IT. Dette reflekteres også i forskning innen lavkodefenomenet (se for eksempel Bock og Frank, 2021; Krejci et al., 2021; Sanchis et al., 2020).

### 2.2.2 Databehandling i skyen og tjenestemodeller

I 2020 benyttet 64 prosent av alle foretak i Norge en eller flere skytjenester (Eggen et al., 2021). Dette er også tilfelle i regnskapsbransjen hvor mange av de mest brukte systemene i dag er skybasert (Baksaas et al., 2019; Regnskap Norge, 2021b).

Amazon Web Services var et av de første selskapene som brukte ordet *sky* i markedsføringen av sitt produkt. Deretter dukket begrepet databehandling i skyen/skytjenester opp (Miyachi, 2018). Selv om ordet sky kan gi assosiasjoner til svevende datasentre og informasjon som flyr rundt på internett, er også databehandling i skyen avhengig av fysiske servere hvor data lagres (Smith, 2011). Datatilsynet definerer skytjenester som en samlebetegnelse på alt fra dataprosessering og datalagring til programvare på servere som er tilgjengelig fra eksterne serverparker tilknyttet internett (Datatilsynet, 2018).

Det er det vanlig å dele skytjenestene opp i ulike tjenestemodeller, hvor to av disse er relevant for denne oppgaven. Programvare som en tjeneste (software as a service – SaaS)

er en metode som innebærer å tilgjengeliggjøre programvare slik at brukeren får tilgang til programmet via internett (Microsoft, u.å.). Normalt kan programvaren brukes med en nettleser på for eksempel en PC, istedenfor at programmet må installeres lokalt på egen PC eller server. Plattform som en tjeneste (platform as a service – PaaS) er en løsning hvor kunden setter opp og har kontroll over egenutviklede eller anskaffede applikasjoner i leverandørens infrastruktur ved å bruke programmeringsspråk og verktøy som støttes av leverandøren (Datatilsynet, 2018). Kunden har fortsatt ikke kontroll over nettverk, servere, operativsystemer eller lagringsmuligheter.

De skybaserte regnskapssystemene er typiske SaaS-løsninger og brukes ofte i sammenheng med andre skybaserte fagsystemer som er tilpasset den enkelte kunde eller bransje (Baksaas et al., 2019). Disse fagsystemene er ofte integrert med regnskapssystemet. Graden av integrasjon kan deles i følgende stadier:

1. ny manuell inntasting
2. partivis overføring
3. speiling i sanntid

En ny manuell inntasting innebærer at en person fysisk må legge inn den samme informasjonen som allerede finnes i et annet system på nytt. Neste stadiet er partivis overføring hvor et parti med informasjon, for eksempel en dag, uke eller måned med salg overføres fra et system til et annet enten av en person, robot eller gjennom integrasjon. Ved speiling i sanntid vil den informasjonen som registreres i et system umiddelbart også registreres i det andre systemet. Prosessen vi senere skal se på er på det første stadiet, manuell inntasting, og ambisjonen er å løfte denne til det siste stadiet, speiling i sanntid.

Casebedriften Økonomihuset bruker det skybaserte regnskapssystemet Poweroffice GO, heretter GO. Systemet består av fem moduler som har speiling av informasjon i sanntid og har per i dag over 300 tredjeparts integrasjoner som enten bruker partivis overføring eller speiling i sanntid (Poweroffice, u.å.b).

### 2.2.3 Dataoverføring mellom systemer

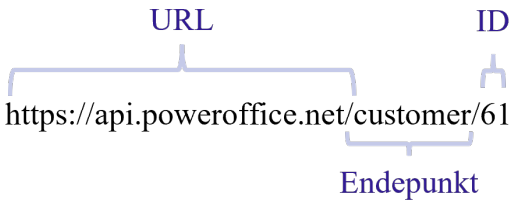
Teknologisk utvikling gjør at bedrifter bruker stadig flere digitale verktøy, og et viktig element i digitaliseringen er å knytte disse systemene sammen. Regnskapsføring handler

om informasjonssamling og strukturering (Gustavsen og Baksaas, 2019). En viktig del av digitaliseringen i bransjen er derfor å samle informasjon på en effektiv måte, og sikre at data som allerede eksisterer i et system ikke må opprettes manuelt på nytt i et annet system. Automatisk dataoverføring mellom ulike systemer kan gjøres på flere måter, blant annet med teknologiene API og RPA.

API (Application Programming Interface) er en innovasjon for å forenkle programvareutvikling. Selv om teknologien for de fleste er usynlig ville hverdagen sett ganske annerledes ut uten (Sandvik, 2019). APIer eksisterer i utgangspunktet for å kunne utveksle informasjon mellom to eller flere programmer. Teknologien stammer fra tiden PC'en ble oppfunnet, men internett APIer ble først tatt i bruk rundt år 2000. I nyere tid er teknologien blitt så viktig at den etterhvert kan regnes som bindeleddet i det digitale økosystemet som knytter sammen virksomheter, programmer og økonomier for å skape verdier og nye muligheter (Ofoeda et al., 2019). Siden de fleste moderne regnskapssystem er skybaserte er API en svært relevant teknologi for bransjen.

API er grensesnittet en må forholde seg til for å hente ut informasjon eller for å få utført en handling i et system. APIer skaper et felles koblingspunkt slik at ulike systemer kan "snakke sammen" (Ofoeda et al., 2019).

Et API består som oftest av en URL med flere ulike endepunkter. URLen er basen og endepunktet styrer hvilken informasjon som hentes eller leveres. Endepunktene er normalt navngitt slik at de korresponderer med informasjonen som ligger bak. I GO sitt API hentes for eksempel kundeinformasjon i endepunktet *customer* (Poweroffice, u.å.a). Videre kan det hentes ut informasjon om en spesifikk kunde ved å referere til ID. Denne oppbygningen vises i figur 2.1.




**Figur 2.1:** Oppbygning av URL (Poweroffice, u.å.a).

Protokollen for å sende og motta virtuelle data på internett er HTTP(S)-forespørsler eller spørringer, hvor *S* representerer en sikkerhetsprotokoll (Hubbard, 2019). Det er mulig å sende en rekke forskjellige forespørsler. Under vises mulighetene i GO:

- GET: For å hente informasjon fra et system.
- POST: For å legge inn informasjon i et system.
- DELETE: For å slette informasjon fra et system.
- PUT: For å oppdatere informasjon (eksempelvis på et eksisterende element) i et system.

En spørring vil alltid levere en respons i et strukturert tekstformat som svar på forespørselen. GO leverer respons i formatet JavaScript Object Notation (JSON) (Poweroffice, u.å.a). Responsen inneholder alltid en status og en tekst. Statusen er en tresifret kode som forteller om forespørselen var vellykket eller om den feilet. Teksten inneholder informasjonen som ble forespurt (i en GET-spørring) og en beskrivelse av forespørselens status. Figur 2.2 viser et eksempel på responstekst fra GO.


```
{
   "data": [
      {
         "name": "Accomodo Regnskap AS",
         "vatNumber": "989746111",
         "id": 1,
         "code": 15591,
      }
   ],
   "success": true,
   "count": 1
}
```


**Figur 2.2:** Responstekst i JSON fra GET-spørring mot GO sitt API-endepunkt: *customer* (Poweroffice, u.å.a).

Forkortelsen RPA står for Robotic Process Automation og er betegnelsen på en virtuell robot som etterligner menneskelig aktivitet ved å utføre standardiserte, repetitive oppgaver basert på et definert sett av regler og forutsetninger (Osmundsen et al., 2019). RPA-verktøy er, i likhet med lavkodeplattformer, laget slik at det enkelt kan settes opp uten tradisjonell koding. Robotene interagerer med eksisterende systemer på samme måte som mennesker

og ingen av systemene må tilpasses for å ta i bruk RPA. Teknologien krever høy grad av vedlikehold, da den baserer seg på at underliggende systemer ikke endres (Stople et al., 2017). På grunn av dette er RPA best egnet i de virksomhetene som har systemer i bunn som er svært dyre å bytte ut eller endre, og som ikke lar seg integrere gjennom for eksempel API (Murphy, 2018).

## 2.3 Lavkode

### 2.3.1 Hva er lavkode?

Konseptet omtalt som lavkode betegner et sett med verktøy for utviklere og personer uten utviklingskompetanse (Waszkowski, 2019). Hensikten er å kunne realisere applikasjonsløsninger raskt, enkelt og smidig. Utviklingen skjer i et dynamisk og visuelt grensesnitt, ofte sammensatt av grafiske dra-og-slipp verktøy. Den tekniske koden er skjult bak funksjonelle byggeklosser, komponenter og moduler som skal gi brukeren et intuitivt miljø å forholde seg til. Konseptet lavkode innebærer at det er rom for eller i noen tilfeller nødvendig med teknisk kode for en tilpasset løsning (Bock og Frank, 2021; Sahay et al., 2020; Sanchis et al., 2020). Dette kan eksempelvis være å tilføre en bedriftsspesifikk betingelse til en byggekloss. Et beslektet konsept omtales som nullkode (no-code) og beskriver en versjon hvor brukeren ikke benytter kodespråk (Cabot, 2020). Det er flere tilbydere som posisjonerer seg selv som en nullkodeplattform i markedet. Per i dag er det som regel likevel mulig å tilpasse applikasjonsutviklingen med kodespråk (Di Ruscio et al., 2022). Det er derfor hensiktsmessig å behandle de to konseptene under ett som lavkode videre i litteraturkapittelet.

Leverandører av lavkodeverktøy tilbyr gjerne programvaren i en skytjeneste, levert som PaaS. På engelsk omtales disse plattformene med forkortelsene LCDP eller LCD for Low-Code Development Platforms (se eksempelvis Bock og Frank, 2021; Sahay et al., 2020; Sanchis et al., 2020). Videre i oppgaven brukes lavkodeplattform som en norsk versjon av begrepet. Lavkodeplattformer tilbyr et miljø for å håndtere hele applikasjonens livssyklus, inkludert utvikling, distribusjon og vedlikehold.

### 2.3.2 Historisk perspektiv

Allerede på 50-tallet ble språket Fortran skapt for å forenkle programmeringsutviklingen (Woo, 2020). Programmering er å sette opp en serie instrukser som avgjør hvordan en maskin skal reagere på brukerens handlinger (Rossen, 2019b). Praktiserende utviklere og akademikere har forsøkt å redusere den tekniske koden som er nødvendig for å produsere et program helt siden verden ble digitisert. På 80-tallet var det fjerde generasjons programmeringsspråk (4GL) og datastøttet programvareteknikk (CASE) (Martin, 1982), på 90-tallet var det Rapid Application Development (Martin, 1991) og på 2000-tallet var det sluttbrukerutvikling (Lieberman et al., 2006). De siste to tiårene har vært preget av modelldrevet prosjektering (Model Driven Engineering, MDE) (for eksempel Da Silva, 2015; Schmidt, 2006). Målet med MDE er å øke produktiviteten og kvaliteten på arbeidet ved å automatisere ulike steg i programvareutviklingen (Di Ruscio et al., 2022). I likhet med lavkode er deler av koden, som for eksempel sikkherhetmekanismer, skjult bak komponenter.

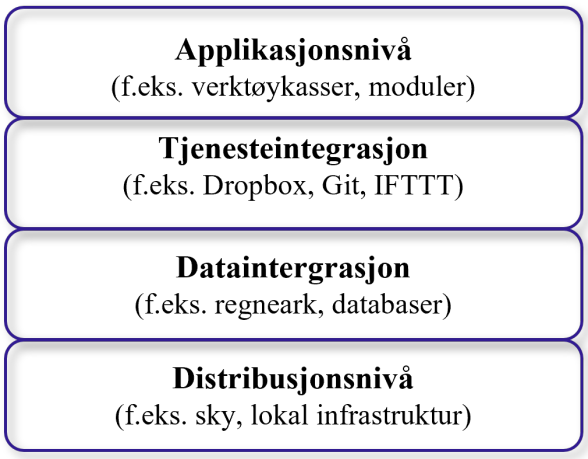
Lavkode som begrep spores som regel tilbake til en markedsanalyse publisert av Forrester i 2014 (Richardson og Rymer, 2014). Her defineres lavkodeplattformer som plattformer som muliggjør rask levering av forretningsapplikasjoner med et minimum av teknisk koding og forhåndsinvestering i oppsett, opplæring og distribusjon (Richardson og Rymer, 2014, s. 2). Forrester definerer senere lavkodeplattformer på nytt som produkter og/eller skytjenester for applikasjonsutvikling som bruker visuelle, deklarative teknikker i stedet for programmeringsspråk (Rymer, 2017). Lavkodeplattformer er tilgjengelig for kunder til lav eller ingen monetær- og opplæringskostnad for å komme i gang. Kostnadene er deretter proporsjonalt økende avhengig av nytteverdien av plattformen for virksomheten (Rymer, 2017).

Begrepet lavkode er ikke vitenskapelig definert (Cabot, 2020). Bock og Frank (2021) hevder at lavkode brukes inkonsekvent og primært for å selge heterogene utviklingsmiljø. Som nevnt er ikke programvareutvikling med redusert kodespråk noe nytt og lavkode er derfor ingen radikal innovasjon (Bock og Frank, 2021). Det er en pågående debatt om hvorvidt eksisterende forskning innen MDE er direkte overførbart til lavkoding. Cabot (2020) skriver at det er mulig å erstatte ordet *modelldrevet* med *lavkode* som et gratis forskningsbasert veikart for lavkodeutvikling. Tilnærmingene er svært like når det kommer

til målsetting og intensjon av teknologien, men konseptene ikke er identiske (Di Ruscio et al., 2022). Ikke alle MDE-tilnærminger søker å redusere teknisk koding og ikke alle lavkodeløsninger er modelldrevet. Ettersom de to står hverandre nært konseptuelt oppstår det likevel et potensiale for å anvende eksisterende kunnskap og krysspollinering mellom de to disiplinene (Di Ruscio et al., 2022). Innenfor rammene av denne masteroppgaven er det hensiktsmessig å fokusere på forskning som spesifikt referer til lavkodeplattformer.

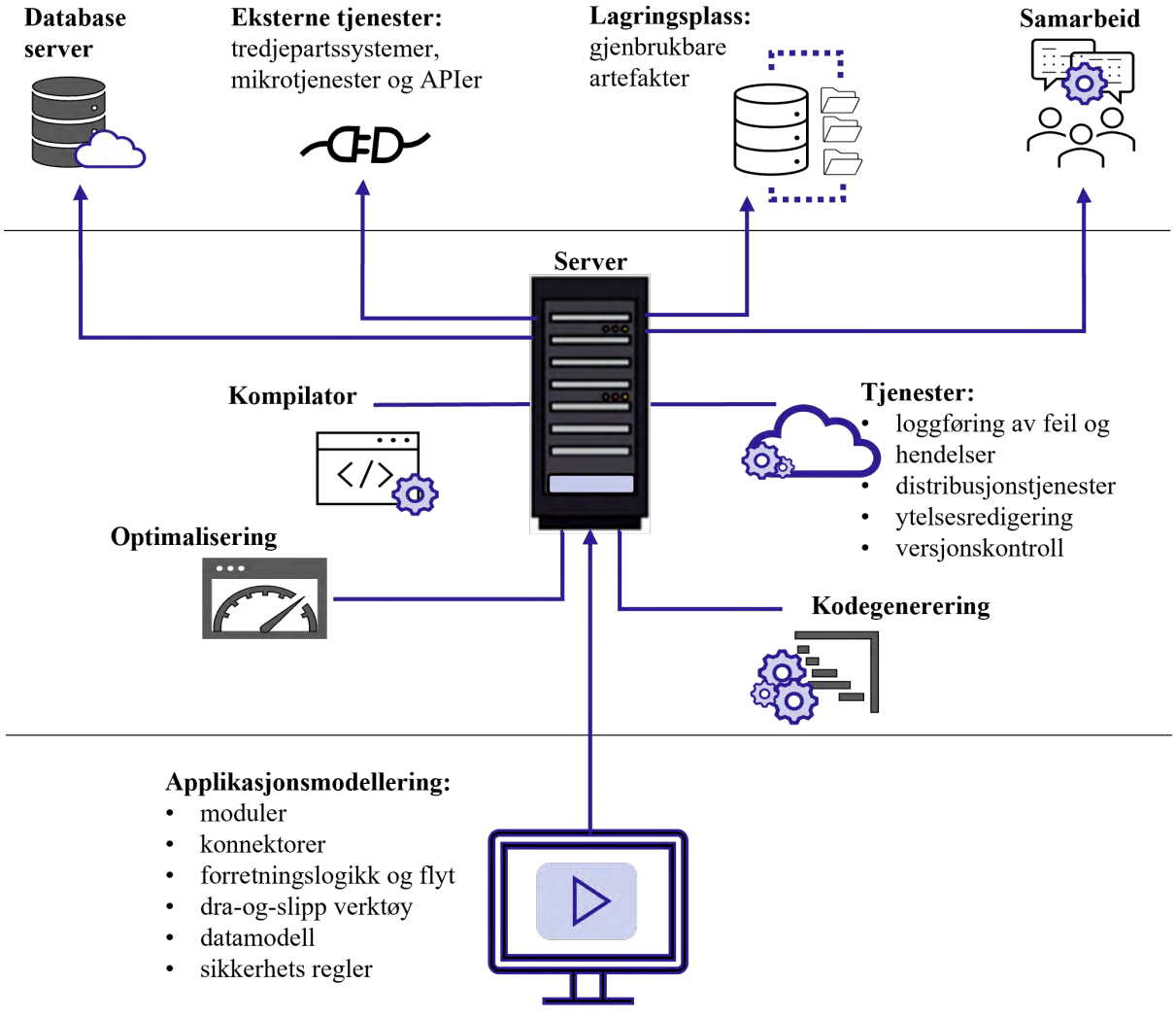
### 2.3.3 En oversikt over lavkodeplattformer

Sahay et al. (2020) analyserer åtte lavkodeplattformer vurdert som representative for å bidra til et konseptuelt komparativt rammeverk. Deres arbeid gir en oversikt over vanlige egenskaper ved arkitekturen og infrastrukturen for plattformene, i tillegg til en typisk utviklingsprosess. De *arkitektoniske* aspektene er delt i fire nivåer. Øverst er applikasjonsnivået som representerer et grafisk brukermiljø. Brukeren har verktøykasser og moduler til å konstruere brukergrensesnittet til en applikasjon. Neste nivå er tjenesteintegrasjon. Dette nivået anvendes til å koble sammen tjenester ved å bruke APIer og autentiseringsmekanismer. Deretter følger nivået for dataintegrasjon som håndterer integrasjon med ulike typer datakilder. Siste nivået i arkitekturen er distribusjonsnivået. Avhengig av leverandør innebærer dette om en applikasjon kan distribueres enten i skyen og/eller lokalt.




**Figur 2.3:** Arkitektoniske nivå for lavkodeplattformer oversatt fra Sahay et al. (2020, s. 3).

En typisk *infrastruktur* for lavkodeplattformer er sortert i tre trinn som illustrert i figur 2.4 (Sahay et al., 2020). I bunn er applikasjonsmodelløren hvor utvikleren lager programmet. Neste trinn består av plattformserveren. Her genereres koden for applikasjonen som er laget og applikasjonen kan testes og produseres. Alle mikroelementene til tjenesteplattformen orkestreres i plattformens backend uten innblanding fra brukeren. På denne måten fritas brukeren ansvaret ved å håndtere tekniske aspekter med for eksempel autentisering og sikkerhet. Det tredje og øverste trinnet innebærer eksterne tjenester som er integrert med plattformen. Det kan for eksempel være arkivering av gjenbrukbare artefakter eller muligheten til å samarbeide i sanntid via skyen.




**Figur 2.4:** Infrastruktur for lavkodeplattformer oversatt fra Sahay et al. (2020, s. 3).

En vanlig *utviklingsprosess* via lavkodeplattformer oppsummeres i fem korte trekk (Sahay et al., 2020; Di Ruscio et al., 2022):

1. Datamodellering: for eksempel definere begrensinger, betingelser og relasjoner.
2. Etablere brukergrensesnitt: for eksempel opprette visninger og visuell data.
3. Spesifisere forretningslogikk og arbeidsflyt: for eksempel fastslå operasjoner for komponentene.
4. Integrere eksterne APIer: for eksempel importere informasjon fra et regnskapssystem.
5. Distribuere applikasjonen.

Di Ruscio et al. (2022) legger også til vedlikehold av applikasjonen som et sjette element i utviklingsprosessen.

### 2.3.4 Motiv for bruk av lavkodeplattformer

Målet med lavkodeverktøy er å fremskynde utviklingsprosessen. Flere artikler finner at utviklingstiden reduseres betraktelig ved bruk av lavkodeplattformer (se for eksempel Bock og Frank, 2021; Sanchis et al., 2020; Waszkowski, 2019). Det er flere elementer som bidrar til hurtig utvikling. Den mest åpenbare er visuell programmering, automatisk kodegenerering og konfigurasjon. Tidsbesparelsen knyttes spesielt opp mot rutineoppgaver ved tradisjonelt utviklingsarbeid (Bock og Frank, 2021). I tillegg reduseres tid som brukes på feilsøking (Waszkowski, 2019). Utvikling og konstruksjon blir også mindre kompleks fordi brukeren ofte kan benytte eksisterende moduler for oppsett. En undersøkelse utført av Forrester viser at lavkodeplattformer akselerer utviklingsarbeidet med opp til 10 ganger (Richardson og Rymer, 2016). Redusert tid brukt på utvikling resulterer i reduserte kostnader.

Skyløsningen muliggjør umiddelbar tilgjengelighet og er en betydelig faktor både for rask utvikling og leveranse av applikasjoner (Di Ruscio et al., 2022). Skybasert utvikling som ikke krever lokal installasjon, senker inngangsbarrieren vesentlig for nye brukere. Brukerne kan evaluere og i noen tilfeller også levere småskala-applikasjoner uten kostnad direkte i egen nettleser. Det er mulig å utvikle enkle produkter for å validere idéer og krav fra kunder uten å kaste bort tid og ressurser på funksjonalitet som kanskje ikke gir verdi til kundene (Richardson og Rymer, 2016).

Programvare er i dag gjennomgripende i alle aspekter av menneskelig aktivitet og etterspørselen etter IT-kompetanse har vokst tilsvarende. Mangel på IT-kompetanse er et motiv for å ta i bruk lavkodeverktøy (se for eksempel Di Ruscio et al., 2022; Metrôlho et al., 2020). Det estimeres at Norge vil ha behov for 40 000 flere sysselsatte med IKT-utdanning innen 2030 (Eggen et al., 2021). Mangelen er størst blant programvare- og applikasjonsutviklere. Mangel på kompetanse berører hele virksomhetens digitaliseringsarbeid og er et vesentlig hinder for utviklingen til norske virksomheter (KPMG, 2020). NHOs medlemsbedrifter melder også at digital kompetanse er en mangelvare, og over halvparten av medlemmene mener generell IKT-kompetanse må heves blant eksisterende ansatte (Næringslivets Hovedorganisasjon, 2021). Anskaffelse av ny teknologi vil også kreve en investering i teknologirelevant kompetanse til å betjene teknologien (Eggen et al., 2021).

Det hevdes at det visuelle grensesnittet til lavkodeplattformer tilrettelegger for at personer uten programmeringsbakgrunn skal være i stand til å produsere og distribuere applikasjoner relativt enkelt (se for eksempel Vincent et al., 2020). Disse refereres til som borgerutviklere (Citizen Developers). Det er en fordel å inkludere ansatte i prosessen med å utvikle løsninger for å dekke behov i deres eget fagfelt (Metrôlho et al., 2020; Sanchis et al., 2020; Woo, 2020; Di Ruscio et al., 2022). Lavkodeplattformer trekkes frem som et lovende utgangspunkt for digital innovasjon med ansatte på tvers av hele organisasjonen (Krejci et al., 2021).

Dagens fasiliteter for opplæring er en motiverende faktor for å ta i bruk lavkodeverktøy (Di Ruscio et al., 2022). Mediene for opplæring har endret seg betraktelig de seneste årene. For noen år tilbake var lærebøker primærkilden til utdanning og opplæring innen utvikling og teknologi. I dag er internett den viktigste kilden, spesielt videodeling via tjenester som YouTube (Di Ruscio et al., 2022). Dette muliggjør et tilnærmet konstant oppdatert opplæringsmiljø rettet mot ulike grupper. Det gir også borgerutviklere en mulighet til å utvikle og dele egne erfaringer og opplæringsmateriell fremfor å være passive forbrukere.

De siste 40 årene har digital kompetanse økt drastisk (Di Ruscio et al., 2022). Grunnleggende programmering undervises nå i flere land som en del av obligatorisk opplæring for nye generasjoner innen ulike fagfelt. NHH har her egne eksempler, som kodespråket R i metodefagene i bachelorprogrammet.

Flere lavkodeplattformer støtter agile arbeidsmetodikker som for eksempel Scrum og Kanban (Sahay et al., 2020). Dette bidrar til bedre teamarbeid (Luo et al., 2021). Til tross for at det er mange motiver for bruk av lavkodeplattformer, bringer teknologien med seg en rekke utfordringer. I neste del presenteres utfordringer som motstrider noen av fordelene som er trukket frem.

### 2.3.5 Utfordringer ved lavkodeplattformer

Skalerbarhet trekkes frem som en utfordring i møte med lavkodeplattformer. Distribuert i skyen skal plattformene helst kunne håndtere stordata, intensive beregninger i høy hastighet, variasjon og volum. Tisi et al. (2019) skriver at lavkodeprogrammer primært brukes til mindre applikasjonsløsninger og påpeker at en evolusjon for storskala og virksomhetskritiske applikasjoner er nødvendig. En tidligere rapport fra Forrester hevder at dette er en myte og viser til eksempler av storskalautvikling (Richardson og Rymer, 2016). Andre studier finner det utfordrende å vurdere, forske og bidra til skalerbarhet på lavkodeplattformene på grunn av mangel på åpne standarder (Sahay et al., 2020).

Fragmentering, manglende standarder og lukkede kilder kan være til hinder for å ta i bruk lavkodeplattformer (Tisi et al., 2019). Leverandørene definerer egne lavkodeparadigmer assosiert med bestemte programmeringsmodeller (Tisi et al., 2019). Ved å benytte lavkodeplattformer oppstår det en risiko for bindingseffekt (Bock og Frank, 2021). Det som er representativt hos en tilbyder kan ikke brukes hos en annen. Dette er en vesentlig trussel mot virksomhetens investering og kan føre til mislykkede prosjekter dersom leverandøren forsvinner eller ikke lengre støtter funksjonalitet som benyttes av virksomheten (Bock og Frank, 2021). Om virksomheten av en eller annen grunn må bytte leverandør medfører det kostnader i form av nedetid, ny utvikling og opplæring (Luo et al., 2021). Ettersom mange lavkodeplattformer er proprietære og har lukkede kilder, vil manglende standarder for interoperabilitet hemme utviklingen av tjenestene generelt (Sahay et al., 2020). Dette begrenser også fasiliteter for læring som ble trukket frem over som et motiv for bruk av lavkodeplattformer.

Et felles økosystem for læring og felles rammeverk eksisterer ikke. Dette er vesentlig for å sikre kontinuerlig vekst og brukerbase blant lavkodeplattformer, spesielt siden leverandørene også markedsfører plattformene mot personer med lav eller ingen

utviklingskompetanse (Di Ruscio et al., 2022). På dette området kan lavkodeplattformene dra lærdom av MDE som har lykkes med et slikt økosystem rundt Eclipse Modelling, hvor rammeverket er etablert både gjennom industrien og akademia (Di Ruscio et al., 2022). En annen konsekvens av manglende standarder er at enkelte plattformer krever omfattende koding for å utvide funksjonalitet. Koden må følge arkitektoniske og designmessige begrensninger ved plattformen (Sahay et al., 2020). Dette setter spesielt begrensninger på erfarne utviklere (Luo et al., 2021).

Til tross for at tilbyderne også markedsfører tjenestene sine mot borgerutviklere finnes det flere artikler hvor resultatet tyder på at kompetanse er en utfordring. Enkelte plattformer har ikke et intuitivt grensesnitt og mangler undervisningsmateriell, eksempler og nettbaserte opplæringsprogrammer (Sahay et al., 2020). Derfor krever noen plattformer utviklingskompetanse. Undersøkelser av Stack Overflow og Reddit viser mange eksempler på brukere som ikke synes lavkodeplattformene er så enkle som det gis uttrykk for og at det derfor kan være bedre å investere i allerede ferdige applikasjoner (Luo et al., 2021). Vedlikehold og feilsøking oppleves spesielt vanskelig for personer uten teknisk kompetanse (Woo, 2020). IT-personell er en nødvendig støtte for borgerutviklere under hele applikasjonens livssyklus, fra idéutveksling og underliggende tekniske forutsetninger til distribusjon (Krejci et al., 2021). Vi har ikke lykkes i å finne forskning på opplæring innen lavkodeplattformer for borgerutviklere. Studier viser imidlertid at lavkode er egnet for personer med allerede teknisk kompetanse og personer med høyere kvalifikasjoner innen andre tekniske fagfelt (Metrôlho et al., 2020).

Uten profesjonelle utviklere kan lavkodeapplikasjoner bli vanskelig å oppdatere for å tilfredsstille tekniske krav (Woo, 2020). Applikasjonens kjøretid kan bli mindre effektiv og det kan bli vanskeligere å integrere applikasjonen med virksomhetens større programvaresystemer. Mangel på inkludering av IT-personell vil kunne utgjøre en risiko for sikkerhet (Woo, 2020; Sanchis et al., 2020). Dersom utvikling via lavkodeplattformer ikke testes og godkjennes av IT-avdelingen kan det føre til skygge-IT (Sanchis et al., 2020). Det betyr skjulte trusler som en vanlig ansatt gjerne ikke har tenkt på eller har noen forutsetning for å identifisere. På den måten vil en virksomhet for eksempel ikke være i stand til å beskytte seg mot at data havner på avveie.

## 2.4 En oppsummering

Litteraturgjennomgangen viser at digitalisering har store konsekvenser for regnskapsførerselskap. Eksisterende forskning fokuserer på rolleendring i selskapene og hva en regnskapsfører skal gjøre i fremtiden (Knudsen, 2020; Marrone og Hazelton, 2019; Schei et al., 2015). Dagens teknologi krever en kompetanse utover det tradisjonelle fagområdet til regnskapsfører (Marrone og Hazelton, 2019) og regnskapsførerens rolle utfordres av andre yrkesgrupper (Hazgui og Gendron, 2015; Arnaboldi et al., 2017a). Det er lite fokus på hvordan selskapene kan ta en aktiv rolle i den tekniske omstillingen. I tillegg fremhever litteraturen et behov for å kommunisere forskningen på en måte som tilrettelegger for å anvende kunnskapen i regnskapsbransjen (Guthrie og Parker, 2016, 2017). Systemleverandørene får stadig mer makt i bransjen (Kornberger et al., 2017) og det kan derfor være nyttig for regnskapsførerselskap å være i stand til å utfordre disse.

Lavkodeverktøy setter flere i stand til å ta del i applikasjonsutvikling (se for eksempel Di Ruscio et al., 2022; Luo et al., 2021) og det estimeres en sterk økning i bruken av teknologien fremover (Gartner, 2021). Forskning viser at det er en fordel å inkludere fagpersoner i utviklingsprosessen av nye løsninger innenfor deres eget fagfelt (Metrôlho et al., 2020; Sanchis et al., 2020; Woo, 2020; Di Ruscio et al., 2022). Det er derfor interessant å undersøke om regnskapsførerselskap kan bruke lavkodeverktøy til å automatisere arbeidsoppgaver og følge den digitale utviklingen. Gjennom en praktisk tilnærming og et konkret case håper vi at studien kan bidra til å sette regnskapsførerselskap i stand til å få fart på digitaliseringen som kreves for å være konkurransedyktig i fremtidens marked.

# 3 Metode

I dette kapittelet redegjør vi for benyttet forskningsmetode. Vi undersøker problemstillingen "*Er lavkodeverktøy egnet til å automatisere arbeidsoppgaver i regnskapsbransjen og bidra til regnskapsførerselskapenes digitale utvikling?*". For å kunne forstå lavkodeverktøy som fenomen, valgte vi å benytte designvitenskapelig forskningsmetode (Design Science Research Methodology, DSRM) og dermed selv utvikle en løsning som automatiserer en reell arbeidsoppgave i et regnskapsførerselskap. Dette for å generere kunnskap gjennom handling. DSRM er en etablert metode innen fagfelt som ingeniørvitenskap og informatikk, men kanskje ikke tradisjonelt brukt ved en handelshøyskole. Teknologi og digitalisering er essensielt i næringsvirksomhet og dermed en stor del av forskningen innen andre disipliner, deriblant økonomi. Det finnes relativt lite forskningsbasert kunnskap om lavkodekonseptet og ingenting om bruk av lavkodekodeverktøy i regnskapsførerbransjen. Vi tror derfor at DSRM kan gi oss en bedre situasjonsbasert forståelse av og kunnskap om anvendelse av lavkodekonseptet i regnskapsbransjen enn andre tradisjonelle metoder. Kapittelet redegjør for hva designvitenskapelig forskningsmetode er og hvordan vi anvender metoden i masteroppgaven.

## 3.1 Designvitenskapelig forskningsmetode

Lavkodetrenden posisjonerer seg som et bidrag til forskning innen informasjonssystemer for bedrifter (Bock og Frank, 2021), og DSRM blir vektlagt som en viktig metodisk tilnærming innen IS-litteraturen (se for eksempel Vaishnavi og Kuechler, 2021; Geerts, 2011; Hevner et al., 2004; March og Smith, 1995). DSRM fremstiller artefakter, det vil si menneskeskapte objekter. I motsetning til naturvitenskap som studerer virkeligheten og hvordan ting *er*, fokuserer designvitenskapelig metode på hvordan ting *burde være* (Simon, 1996).

Forskningsmetoden omfatter to primære aktiviteter: 1) nyskapning gjennom design av nye eller innovative artefakter og 2) analysere artefaktens bruk og/eller ytelse (Vaishnavi og Kuechler, 2021). Forskningsprosessen inkluderer, men er ikke begrenset til, algoritmer, grensesnitt for mennesker og datamaskiner og metoder for systemdesign eller språk.

Tilnærmingen kan adressere et problem på en unik og innovativ måte eller på en mer effektiv måte (Hevner et al., 2004).

Metoden er normativ og teknologiorientert hvor utfallet evalueres i forhold til verdi og nytte (March og Smith, 1995). Det stilles to krav for å oppnå nytteverdi i designvitenskapen, relevans og rigorøsitet (se for eksempel Hevner et al., 2004). Relevans forekommer når artefakten løser et reelt forretningsproblem. Rigorøsitet, som kommer av det engelske ordet *rigor*, oppnås når eksisterende kunnskap, for eksempel teoretiske grunnlag og metoder, anvendes.

### 3.1.1 Forventinger til resultat

Resultatet av et designvitenskapelig forskningsprosjekt bør være kunnskap (Vaishnavi og Kuechler, 2021). Kunnskapen skal tilføre noe nytt, relevant og interessant for det aktuelle publikum. March og Smith (1995) skiller mellom fire ulike typer artefakter: konsepter, modeller, metoder og eksemplarer. I senere tid inkluderes også teori som en type artefakt (se for eksempel Gonzalez og Sol, 2012; Cleven et al., 2009). I denne masteroppgaven designer vi et *eksemplar*. Et eksemplar realiseres i et miljø som benytter konsepter, modeller eller metoder. Å designe et eksemplar kan i noen tilfeller overgå fullstendige modeller eller konsepter som eksemplaret er en del av (March og Smith, 1995). Det kan for eksempel tenkes at forståelsen som opparbeides om fenomenet ikke ville eksistert med mindre eksemplaret ble skapt. Resultatet av å designe et eksemplar er derfor *situasjonsbasert anvendelse* av eksisterende konsept, metode eller modell (Vaishnavi og Kuechler, 2021).

En konstruksjonsprosess er et sett med trinn som brukes til å utføre en oppgave (Vaishnavi og Kuechler, 2021). I designvitenskapelig forskning kan en konstruksjonsprosess i seg selv være gjenstand for forskningsprogrammet (Cleven et al., 2009). For at vi skulle være i stand til si noe om nytten av lavkodeverktøy ønsket vi å erfare prosessen i praksis gjennom et reelt problem.

## 3.2 Rammeverk

Vi anvender designvitenskapelig forskningsmetode som presentert av Peffers et al. (2007). Geerts (2011) har laget en oversiktlig tabell av dette rammeverket som vi benytter og illustrerer i tabell 3.1. Den første kolonnen i tabellen viser DSRM som en sekvens av seks

aktiviteter. Den andre kolonnen beskriver aktivitetene, mens den tredje kolonnen knytter kunnskapsgrunnlag til aktivitetene. Pilene til venstre i tabellen illustrerer iterasjon som en del av forskningsmetoden. Det betyr at aktiviteter som *Evaluering* og *Kommunikasjon* ofte resulterer i en revidering av artefaktenes mål og design.

| DSRM: AKTIVITETER | AKTIVITETS-BESKRIVELSE | KUNNSKAPSGRUNNLAG |
|---|---|---|
| **Identifisering av problem og motivasjon** | *Hva er utfordringen?*<br><br>Regnskapsbransjen må endre seg i takt med den digitale utviklingen. Kan lavkodeverktøy bidra til denne omstillingen? | Regnskapsførerrollen er i endring som et resultat av økt digitalisering. Norge mangler IT-kompetanse. Eksisterende forskning fokuserer i liten grad på hvilke teknologi som skal benyttes i den digitale omstillingen. |
| **Definere mål med løsningen** | *Hvordan skal utfordringen løses?*<br><br>Realisert løsning skal automatisere en arbeidsprosess og eliminere tidkrevende og unødvendige arbeidsoppgaver. | Kunnskap om eksisterende løsning for å sette oss i stand til å utvikle et bedre alternativ. |
| **Design og utvikling** | *Lag en artefakt som løser utfordringen.*<br><br>Realisere en applikasjon ved bruk av lavkodeverktøy. | Kunnskap om lavkodeverktøy som konstruksjonsprosess. |
| **Demonstrasjon** | *Demonstrer bruken av artefakten.*<br><br>Demonstrerer applikasjonen ved å vise til funksjonalitet. | Kunnskap om applikasjonen tilfredsstiller en forbedring av dagens prosess. |
| **Evaluering** | *Hvor godt fungerer artefaktet?*<br><br>Evaluering av lavkodeverktøy og eksemplaret diskuteres gjennomgående i kapittel 4 og 5 av masteroppgaven. | Definere kriterier for vellykket artefakt. Definere prestasjonsområder for å evaluere lavkodeverktøy. |
| **Kommunikasjon** | Konklusjoner og refleksjoner ved DSRM kommuniseres gjennom masteroppgaven. | Kunnskap om fagkultur og DSRM. |

**Tabell 3.1:** Designvitenskapelig forskningsmetode oversatt og anvendt (Geerts, 2011, s. 144).

Validitet i DSRM avhenger av underliggende filosofisk grunnlag (Gonzalez og Sol, 2012), altså hvordan vi forholder oss til hva som er virkelighet. Vi benytter et rammeverk av Cleven et al. (2009) for klassifisering av evalueringsstruktur for å underbygge validitet. Dette rammeverket består av en rekke variabler og tilhørende verdier som klassifiserer filosofisk grunnlag og hvilke type slutninger som benyttes. Ved å anvende dette rammeverket gir vi både leseren og andre designvitenskapelige bidragsytere muligheten til å sammenligne forskningsinnsatsen basert på de oppgitte variablene. Cleven et al. (2009) konkluderer med at evaluering ikke bør ansees som en isolert prosess, men heller som en del av designprosessen, altså metodeaktivitetene. Derfor redegjør vi for evalueringsstruktur og de ulike variablene med respektive verdier under hver metodeaktivitet. Det er likevel hensiktsmessig å illustrere rammeverkene hver for seg, henholdsvis i tabell 3.1 og tabell 3.2. I tabell 3.2 er verdiene som anvendes i masteroppgaven uthevet.

| VARIABLER | VERDIER | | | | |
|---|---|---|---|---|---|
| Tilnærming | **Kvalitativ** | Kvantitativ | | | |
| Artefaktfokus | Teknisk | **Organisatorisk** | Strategisk | | |
| Artefakttype | Konsepter | Modeller | Metoder | **Eksemplar** | Teori |
| Epistemologi | Positivisme | **Interpretivisme** | | | |
| Funksjon | Kunnskapsfunksjon | **Kontrollfunksjon** | Utviklingsfunksjon | Legitimeringsfunksjon | |
| Metode | Handling | Case | Feltforskning | Formelle bevis | |
| | Kontrollert eksperiment | **Prototype** | Målundersøkelse | | |
| Objekt | **Artefakt** | **Artefakt konstruksjon** | | | |
| Ontologi | Realisme | **Nominalisme** | | | |
| Perspektiv | Økonomisk | Implementering | **Prosjektering** | Epistemologisk | |
| Posisjon | Ekstern | **Intern** | | | |
| Referansepunkt | Artefakt mot forskningsgap | **Artefakt mot reelle problem** | Forskningsgap mot reelt problem | | |
| Tid | Ex ante | **Ex post** | | | |

**Tabell 3.2:** Evalueringsstrategi oversatt fra Cleven et al. (2009, s. 4).

### *Identifisering av problem og motivasjon*

Den første aktiviteten i rammeverket handler om å identifisere et problem og redegjøre for motivasjonen til å undersøke problemet. Idéen for masteroppgaven kom av en introduksjon til lavkodekonseptet og lovnadene som markedsføres av tilbyderne. Lavkodekonseptet som fenomen er interessant og relevant for norsk næringsliv dersom det tilfredsstiller behovene for hurtig applikasjonsutvikling. Ettersom en av studentene er ansatt i Økonomihuset diskuterte vi spesielt regnskapsførerrollen og dens utsikter. Som beskrevet i kapittel 2.1 er regnskapsførerrollen særlig utsatt for digitalisering. IT-verktøy er avgjørende for kostnadseffektive regnskapstjenester. I denne oppgaven ønsker vi å se nærmere på hvorvidt lavkodeverktøy kan tjene regnskapsbransjen. Fagkompetanse sammen med teknisk kompetanse kan sette ansatte i stand til å tilfredsstille enkle og umiddelbare digitale behov internt og for kunder. Dette kan være lønnsomt dersom det effektiviserer arbeidsprosesser og frigjør tid til mer verdiskapende aktiviteter - først og fremst for regnskapsførerselskapet.

I samtaler med Økonomihuset kom vi frem til en prosess vi vurderte som passende for oppgavens natur og representativ i forhold til kompleksitet og hovedelementer mellom systemer som benyttes av Økonomihuset. Prosessen utdypes i kapittel 4.1 Dagens prosess.

Som del av evalueringsstrukturen bør vitenskapsfilosofi erklæres i første aktivitet sammen med artefakttype og fokus, objekt, metode og referansepunkt (Cleven et al., 2009). Vi designer en prototype for å effektivisere en prosess. Vi har allerede redegjort for at artefakten er et eksemplar og at referansepunktet er mot den virkelige verden, altså et reelt problem. I tillegg har vi vært inne på variabelen objekt som refererer til om artefakten eller konstruksjonsprosessen er gjenstand for studien. Her vil vi argumentere for at begge deler inngår i vår problemstilling. Utviklingen på lavkodeplattformen er konstruksjonsprosessen og applikasjonen er artefakten. For å vurdere konstruksjonsprosessen må vi også realisere og vurdere artefakten.

En artefakt kan støtte en bedriftstransformasjon på et teknisk, organisatorisk eller strategisk nivå (Cleven et al., 2009). Denne variabelen adresserer i hvilken kontekst artefakten skal benyttes og ikke utviklingskonteksten. Vi vurderer fokus for artefakten på et organisatorisk nivå da vi evaluerer lavkodekonseptet som konstruksjonsprosess for å endre på organisatoriske prosesser i et regnskapsførerselskap.

Epistemologi, også kjent som erkjennelsesteori, er læren om kunnskap og innsikt (Holmen,

2021). Dette er vitenskapsfilosofi som sier noe om hvordan våre egenskaper påvirker prosessen for kunnskapsinnhenting (Cleven et al., 2009). Her skilles det mellom positivisme og interpretivisme. Mens en positivist objektivt iakttar et fenomen, har en interpretivist en fortolkende holdning hvor evaluering av artefakter avhenger av samspill og kontekst mellom forsker og artefakt (Vaishnavi og Kuechler, 2021). Vi inntar en interpretivistisk rolle i denne sammenheng

Ontologi representerer også en del av vitenskapsfilosofien og innebærer hvordan vi oppfatter virkelighet, hva som eksisterer og hvordan noe finnes. Her skilles det mellom to motsetninger, realisme og nominalisme. Mens realisme innebærer at den ytre virkelighet eksisterer uavhengig av vår bevissthet, tolkes nominalisme intersubjektivt (Cleven et al., 2009). Det vil si at flere subjekter, for eksempel en gruppe mennesker, kommer frem til en tilnærmet felles forståelse av virkeligheten. Vi inntar en en nominalistisk tilnærming i oppgaven.

### *Definere mål med løsningen*

I andre fase av forskningsprosessen skal det redegjøres for generelle mål, gjennomførbarhet og ytelse, i tillegg til kriterier for løsningen. Gjennom samtaler med Økonomihuset har vi kommet frem til følgende kriterier som må oppfylles for at applikasjonen skal være vellykket:

- automatisere prosessen og fjerne repeterende oppgaver for regnskapsfører
- eliminere menneskelige feil
- ha samme funksjon for klientene
- være det foretrukne valget for klientene

For evalueringsstruktur skal det i denne metodeaktiviteten redegjøres for tilnærming, funksjon og perspektiv. Vi benytter et prosjekteringsperspektiv da vi ønsker å vurdere om lavkodeverktøy egner seg til å automatisere arbeidsprosesser i et regnskpsførerselskap. Evalueringen vår bærer preg av alle funksjonene i oversikten til Cleven et al. (2009), men er først og fremst av en kontrollerende art. Dette fordi vi evaluerer artefakten og konstruksjonsprosessen mot kriterier og prestasjonsområder. Tilnærmingen er kvalitativ hvor kildene til informasjon om prosesser i Økonomihuset er basert på samtaler, mens

kunnskap om lavkodekonseptet er basert på erfaring og observasjoner.

### *Design og utvikling*

Design og utvikling er en kreativ fase i sekvensene av metodeaktiviteter. Denne fasen er beskrevet i kapittel 4.3 Utviklingsprosess. Aktiviteten innebærer prøving og feiling etter hvert som vi blir bedre kjent med lavkodeplattformen for å realisere en applikasjon.

Denne fasen er ment for å gi oss kunnskap om lavkodeverktøy som konstruksjonsprosess. Den genererte kunnskapen skal sette oss i stand til å evaluere hvorvidt lavkodeverktøy løser utfordringer slik leverandørene hevder. På den måten kan vi trekke konklusjoner om hvorvidt verktøyet kan benyttes i regnskapsførerselskap. Derfor kategoriserer vi variabelen Tid som Ex post. Det vil si at vi evaluerer lavkodeverktøyet i etterkant av utviklingen.

### *Demonstrasjon*

I kapittel 4.4 Produkt illustrerer vi funksjonaliteten i applikasjonen vi har laget. I appendiks A1 finnes det innloggingsinformasjon for å teste applikasjonen. I appendiks A2 og A3 viser vi applikasjonen og forklarer hvordan den brukes.

### *Evaluering*

Denne aktiviteten er ment for å vurdere og måle hvor godt artefakten egner seg til å løse den aktuelle utfordringen. Aktiviteten bærer derfor preg av alle variablene i evalueringsstrukturen. I evalueringsprosessen får vi innspill fra Appfarm og Økonomihuset, men i hovedsak evaluerer vi artefakten og konstruksjonsprosessen selv fra en intern posisjon. Som illustrert i tabell 3.1 er evalueringsfasen utsatt for iterasjoner. Gjennom arbeidet med masteroppgaven gjøres det en rekke mikroevalueringer som påvirker resultatet. Flere av mikroevalueringene fremkommer i kapittel 4.3 Utviklingsprosess. Evalueringen av det ferdige produktet er beskrevet i kapittel 4.5 Evaluering.

Litteratur som sammenligner lavkodeplattformer identifiserer prestasjonsområder og vurderer måloppnåelsen på disse (se for eksempel Sahay et al., 2020). For å gjøre det mulig for andre å senere etterprøve eller eller sammenlikne studien identifiserer vi også slike fokusområder. Lavkodeplattformen vi benytter for denne oppgaven vurderes på følgende

fire områder:

- Hvor gode er opplæringsressursene?
- Hvor intuitiv er plattformen?
- Hvor enkelt er det å feilsøke?
- Hvor fleksibel er plattformen?

For konkrete holdepunkter rangerer vi disse spørsmålene på en skala fra 1-5 hvor 1 er svært dårlig og 5 svært god. Denne fremgangsmåten bidrar til en viss stabilitet i målinger selv om teknologien er under utvikling.

*Kommunikasjon*
Siste aktivitet i rammeverket er å kommunisere problemstillingen, løsningen, nytteverdien og graden av nyskaping. Vi bruker masteroppgaven for denne aktiviteten. Vi argumenterer og konkluderer med vårt kunnskapsbidrag i kapittel 6.

## 3.3 Datainnsamling

### 3.3.1 Data fra Økonomihuset

For å innhente informasjon om arbeidsprosesser i Økonomihuset har vi benyttet oss av åpne uformelle samtaler, men med en klar agenda: innsikt og forståelse av aktuelle prosesser. På grunn av oppgavens natur mente vi det var hensiktsmessig med samtaler fremfor eksempelvis et intervju. For at vi skulle kunne benytte en prosess til å designe en prototype måtte den tilfredsstille ulike kriterier. Prosessen måtte være mulig å automatisere og måle. I tillegg burde den være representativ for systembruk i bransjen. Vi presenterte disse kriterene for Økonomihuset som kom med flere forslag til aktuelle prosesser. Vi snakket deretter om omfanget i de ulike alternativene og ble sammen enig om den prosessen vi mente var best egnet for masteroppgaven. Videre brukte vi tid på å sikre at vi forstod alle aspektene av den valgte prosessen ved å stille nødvendige oppfølgingsspørsmål og ha en kontinuerlig dialog gjennom utviklingsprosessen. For å kunne fremstille økonomiske data har vi benyttet informasjon fra Økonomihuset som er basert på sanne premisser.

### 3.3.2 Datakvalitet

Evalueringen i designprosessen er basert på intersubjektive sannheter, personlige tolkninger og refleksjoner på et gitt tidspunkt. Dette har konsekvenser for resultatet av arbeidet. I forskningsmetoder som tradisjonelt anvendes i samfunnsvitenskapen er det blant annet krav til datakvalitet i form av validitet og reliabilitet.

En vanlig forståelse av validitet er at det betegner en sannhet eller virkelighet (Gonzalez og Sol, 2012). I DSRM kan ikke teorier bekreftes som sanne, men som midlertidig gyldig (Lee og Hubona, 2009). Det betyr at dersom forskning endrer en virkelighet i dag, kan den senere bli endret på nytt. Validering av en artefakt bør derfor anses som foreløpig, fremfor sann eller usann (Gonzalez og Sol, 2012). Ettersom designvitenskap er en iterativ metode og artefakter bygger på hverandre kreves det gjerne en innsats over tid fra fagmiljøet for å komme nærmere en virkelighet (Vaishnavi og Kuechler, 2021). Det er derfor viktig å tydeliggjøre begrensninger, mangler og usikkerhet knyttet til kunnskapsbidraget slik at det kan benyttes i videre forskning (Gonzalez og Sol, 2012). Det er også derfor vi redegjør for filosofisk perspektiv i metodeaktivitetene. For informasjonssystemer presenteres det to typer validitet: formativ og summativ (Lee og Hubona, 2009). Formativ validitet er en vurderingsform med fokus på forbedringspotensial fremfor feil og mangler (Gonzalez og Sol, 2012). Summativ validitet innebærer bedømming eller rangering (Helle, 2020). Vi trekker konklusjoner basert på kriterier og subjektiv innsikt som resulterer i en summativ validitet.

Reliabilitet handler om hvorvidt målinger er stabile og konsistente (Svartdal, 2020). Ettersom data om lavkodeverktøy er innhentet på et gitt tidspunkt og tjenesten er under konstant utvikling kan vi ikke konkludere med at dataene er reliable. I tillegg er observasjonene og refleksjonene subjektive. Dersom andre studenter eller fagpersoner anvender samme designvitenskapelige tilnærming som oss i denne situasjonen kan de muligens konkludere annerledes. Trolig er dette årsaken til at reliabilitet ikke diskuteres eksplisitt i DSRM. I kapittel 3.3.3 utdyper vi vår bakgrunn og våre begrensninger som er førende for datainnsamling og kvalitet for lavkodekonseptet.

Dataene om Økonomihuset kan kategoriseres som reliable og valide. En av studentene er ansatt i casebedriften og datainnhentingen kan være påvirket av dette. Vi har lagt

opp samtalene slik at de ikke er preget av eksisterende forhold og kunnskap.På grunn av prosessens natur betrakter vi det som mulig, men usannsynlig, at samtalepartnere besvarer spørsmål strategisk, misforstår spørsmål eller oppfatter spørsmålene som ledende i avgjørende forstand. Det er rimelig å anta at nyanser og aspekter ved prosessene i Økonomihuset kan ha blitt oversett. Dermed kan de utgjøre en kilde til svekket reliabilitet. Det er heller ikke umulig at disse prosessene på et tidspunkt endres. Likevel er dette arbeid som utføres på daglig basis. Samtalepartnerne gir en beskrivelse av en prosess med lite variasjon. Prosessen er derfor representativ og gyldig for Økonomihuset. I diskusjonskapittelet drøfter vi hvorvidt lavkodeverktøy kan benyttes i bransjen generelt. Vi kan ikke uten videre generalisere bedriftsspesifikke prosesser. Det er fornuftig å anta at arbeidspraksis varierer mellom regnskapsbedrifter. Samtidig er bransjen underlagt samme krav og lover, anvender lignende systemer og samler, bearbeider og rapporterer regnskapsinformasjon etter samme standard. Dermed er det ikke urimelig å drøfte lavkodekonseptet generelt for bransjen.

### 3.3.3 Subjektive begrensinger

Våre egenskaper og bakgrunn er førende for besvarelsen av vår problemstilling. Dette er første gang vi benytter oss av designvitenskapelig forskningsmetode. Vi har etter beste evne forsøkt å tolke metoden på en strukturert og riktig måte. Vår oppfatning er at DSRM er omfattende, spesielt i forhold til å forsvare rigorøsitet og validitet. Nyanser i tolkningen av de ulike metodeaktivitetene og evaluringsstruktur var utfordrende og kan potensielt utgjøre en feilkilde. Knapphet på tid er et viktig element som begrenser evalueringsfasen spesielt. Dette hemmer antall iterasjoner. Vi har ikke har rukket å gjennomføre en test for klientene til Økonomihuset, noe som ville ha løftet evalueringen.

Vi er masterstudenter med hovedprofil i Business Analytics. Vi har ikke omfattende teknisk kompetanse i forkant av prosjektet, men er likevel kjent med andre lavkodeverktøy og programmeringsspråk som Python og R gjennom emner på skolen. Derfor har vi en viss teknisk kompetanse, spesielt i forhold til programmeringslogikk. Vi kjenner for eksempel godt til for-løkker og hvis-betingelser. Dette er en fordel når vi bruker lavkodeverktøyet og påvirker vår oppfatning av plattformen. Vi tar fatt på prosjektet med egne antagelser og fordommer om hva lavkodeplattformer er og skal være, bevist og ubevist. Å la plattformen tale for seg selv og gjøre seg selv til kjenne gjennom erfaring er derfor ikke upåvirket.

# 4 Analyse

I dette kapittelet presenterer vi prosessen og resultatene av å følge DSRM-rammeverket. Målet er å undersøke hvorvidt lavkodeplattformer er egnet for automatisering av regnskapsprosesser. Vi starter med å beskrive dagens prosess i Økonomihuset. Deretter presenterer vi lavkodeplattformen vi benytter. Kapittel 4.3 inneholder en detaljert beskrivelse av vår utviklingsprosess, hvordan vi har tenkt og jobbet med utvikling av applikasjonen. Det ferdige produktet demonstreres i Kapittel 4.4. Til slutt evaluerer vi både applikasjonen vår som eksemplar og lavkodeplattformen som konstruksjonsprosess opp mot kriteriene som er satt.

## 4.1 Dagens prosess

Prosessen vi tar for oss er innhenting av informasjon fra kunder for å kunne beregne og utbetale riktig lønn til deres ansatte. Denne informasjonen kaller vi for lønnsgrunnlag.

Det er stor forskjell på adopsjon av teknologi mellom de ulike sluttkundene i regnskapsbransjen (Baksaas et al., 2019). Dette gjelder også for Økonomihuset sine kunder. De mest digitaliserte kundene benytter allerede etablerte metoder for automatisk deling av lønnsinformasjon, slik som for eksempel en integrasjon mellom lønnssystemet og et skybasert timeføringssystem. Enkelte kunder har imidlertid utdaterte og lite effektive rutiner, disse velger gjerne å levere lønnsgrunnlaget for sine ansatte på en mindre effektiv måte, for eksempel i en e-post. De ansatte i Økonomihuset må da legge denne informasjonen inn i kundens lønnssystem manuelt. Det er denne prosessen vi søker å forbedre ved å benytte lavkodeverktøy.


**Figur 4.1:** Illustrasjon av dagens prosess.

Figur 4.1 viser de ulike elementene i en slik manuell prosess. Manuell inntasting av lønnsinformasjon er særlig tidkrevende. Det er også denne delen av prosessen som utløser behovet for en ekstra kontroll hos kunden. Dersom det oppdages feil må prosessen startes på nytt ved at kunden informer regnskapsfører om feilene. For å unngå dette ønsker vi å eliminere den manuelle inntastingen. Da reduseres regnskapsførers tidsbruk og risiko for feil.

En forbedring av denne prosessen kunne vært løst på ulike måter ved bruk av ulike teknologier, for eksempel RPA. Som det fremgår av problemstillingen skal vi undersøke om lavkodeverktøy egner seg til slike problemstillinger i regnskapsbransjen.

## 4.2 Programvarevalg

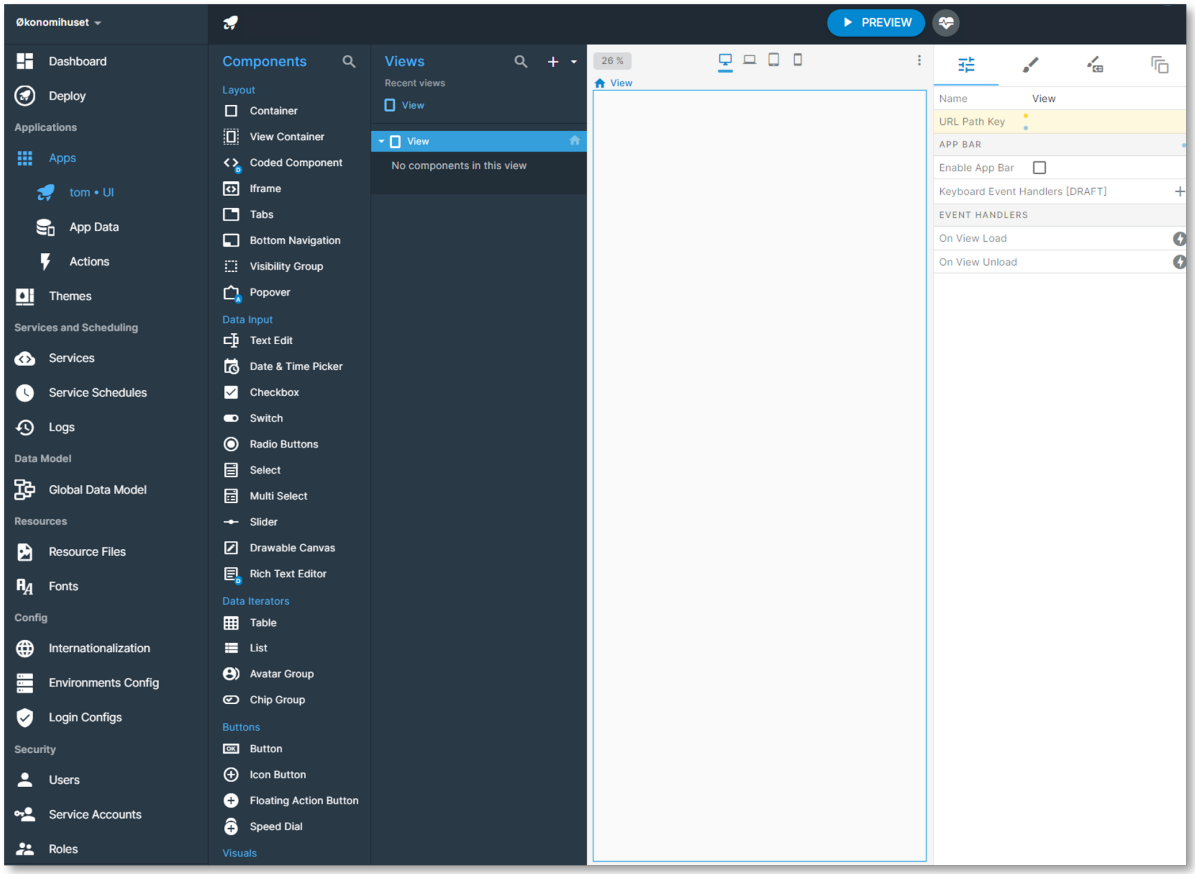
Det finnes i dag hundrevis av leverandører som tilbyr lavkodetjenester. Under arbeidet med masteroppgaven har vi samarbeidet med Avo Consulting AS, heretter Avo, som spesialiserer seg på problemløsning ved bruk av lavkode. Gjennom selskapet ble vi presentert for to alternativer, OutSystems og Appfarm.

En stor fordel med Appfarm er muligheten for skybasert samarbeid i sanntid. Det betyr at vi kan benytte utviklingsmiljøet samtidig, på hver vår datamaskin, også om vi sitter på ulike steder. For eksempel kan vi til enhver tid se hvor den andre er i ferd med å gjøre endringer i applikasjonen. Endringene blir umiddelbart synlig for alle som er innlogget i miljøet. Dette bidrar til enkelt og fleksibelt samarbeid i tjenesten. Det er også relevant i en hverdag med stadig større utstrekning av fleksibelt kontorsted. Til tross for at Appfarm markedsføres som et nullkodeverktøy, kreves det foreløpig noe mer tradisjonell koding i forhold til OutSystems. Dette gjelder for tilpassede applikasjonsløsninger. Med vår bakgrunn vurderte vi dette som gjennomførbart dersom det skulle bli nødvendig og valgte derfor Appfarm.

### 4.2.1 En introduksjon til Appfarm og Appfarm Create

I denne delen gir vi en kort introduksjon og oversikt over Appfarm og Appfarm Create. Appfarm er et norsk selskap som ble etablert i 2017. Selskapet tilbyr tjenester som kan brukes til å utvikle et bredt utvalg av nettapplikasjoner. Selskapet markedsfører seg selv som en nullkodeplattform, men tydeliggjør muligheten for å bruke tradisjonelt kodespråk.

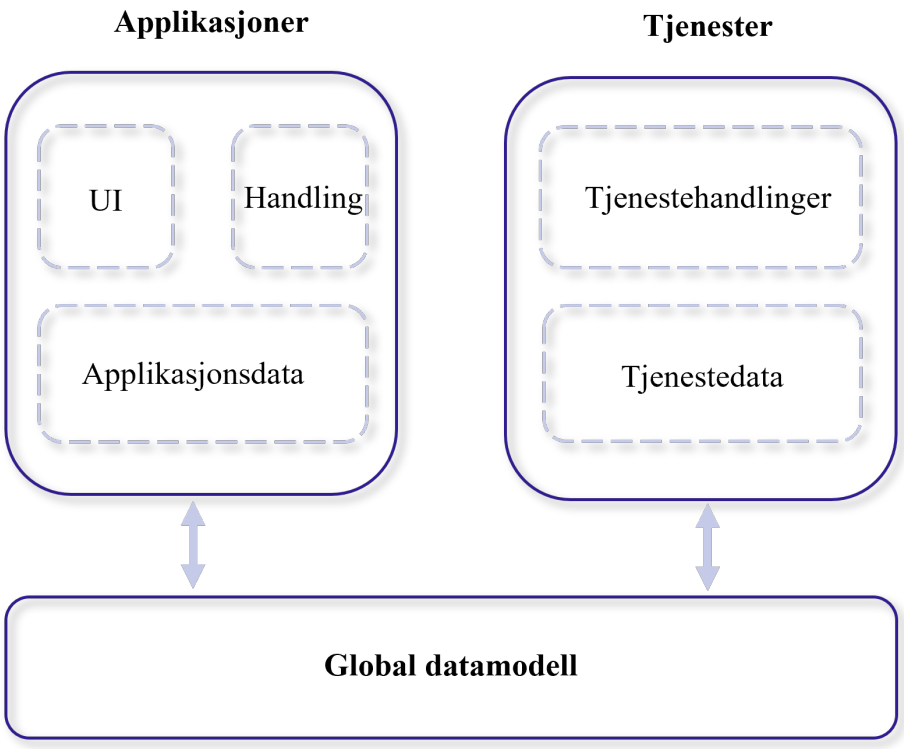
Kodespråket som kan benyttes er JavaScript. Når en løsning utvikles i Appfarm generes det er beskrivelse i JSON. Ved innlasting av applikasjonen i brukerens nettleser fungerer Appfarm sin underliggende teknologi som en leser for JSON beskrivelsen og applikasjonen generes i sanntid.




**Figur 4.2:** Skjermbilde av utviklingsmiljøet Create.

Appfarm Create, heretter Create, er utviklingsmiljøet til Appfarm. Ved innlogging befinner vi oss i en løsningskontekst som vist i figur 4.2. Som Appfarm-utvikler kan en ha tilgang til flere løsninger. Vi har tilgang til en løsning som vi kaller Økonomihuset. Dette er et utviklingsmiljø som inneholder applikasjoner og tjenester som deler en global datamodell. Datamodeller er en visuell representasjon av en database. I den globale datamodellen opprettes det objektklasser som er spesifikt knyttet til virksomheten og funksjonaliteten til applikasjonen som utvikles. For å sette dette i en kontekst som er lettere å forstå, vil det bak det visuelle grensesnittet ligge en tabell. Utvikleren kan opprette kolonner og definere hvilke typer objekter datamodellen skal inneholde. Det kan for eksempel være tekst eller heltall. Figur 4.3 viser hvordan den globale datamodellen er gjensidig avhengig

av tjenestene og applikasjonene som utvikles. I vår oppgave utvikler vi en applikasjon og det er derfor innholdet i ruten til venstre, Applikasjoner, som er relevant.




**Figur 4.3:** En visuell representasjon av nøkkelkonsepter i Create. Hentet og oversatt fra Appfarm (2022).

Applikasjonene i en løsningen har en felles URL-rot for sluttbrukertilgang. På den måten kan en utvikler skreddersy ulike applikasjoner for ulike roller og brukstilfeller. Over datamodellen er det et sikkerhetslag som kontrollerer hvem som kan opprette, lese, oppdatere og slette data gjennom applikasjonene. Brukere kan deles opp i to hovedkategorier: 1) utviklere som har tilgang til Create og 2) brukere som kun har tilgang til applikasjoner opprettet i Create, altså sluttbrukeren. For å kunne åpne en applikasjon må vedkommende få tilstrekkelig tillatelser. Tilgangen kan konfigureres av utvikler. Sikkerhetstrusler og oppdateringer styres av Appfarm og er derfor ikke noe utviklere trenger å ta hensyn til.

UI står for User Interface og er brukergrensesnittet som benyttes av utviklerne. Alle delene av utviklingen kan håndteres visuelt, fra grensesnittet for sluttbrukeren til infrastruktur. For visuell utvikling har Create en verktøykasse med dra-og-slipp verktøy, byggeklosser og komponenter. Det er også mulighet for å utvikle eget design ved bruk av funksjoner

og kodelinjer. I utviklingsmiljøet finnes der en funksjon for å forhåndsvise applikasjonen underveis slik at utviklere kan teste funksjonalitet som er lagt til.

I nivået for handlinger kan en blant annet definere flyt, manipulere og iterere over data, opprette betingelser og navigering i applikasjonene, importere og eksportere filer og sette opp HTTP(S)-forespørsler til eksterne APIer. En handling opprettes ved å kombinere handlingsnoder som består av logiske byggeklosser fra verktøykassen.

For hver Applikasjon kan utvikleren definere hvilke deler av datamodellen som skal benyttes ved å opprette datakilder og velge kardinalitet. I figur 4.3 er dette illustrert som Applikasjonsdata. I datakildene er det mulig binde sammen data, filtrere og sette opp begrensinger.

En løsning kan distribueres på tvers av fire miljøer i skyen – *Develop*, *Test*, *Staging* og *Production*. Vi jobber i utviklingsmiljøet (Develop) hvor endringer blir implementert fortløpende. Test-miljøet brukes til test av applikasjonen for andre enn utviklerne. Staging er siste steg for kvalitetssikring før applikasjonen settes i produksjon. Det som skiller de ulike miljøene er hvilken data som benyttes og URLen. På grunn av dette kan for eksempel utvikling gjøres parallelt selv om en versjon av applikasjonen er i produksjon.

Som en sammenligning til Sahay et al. (2020) sitt arkitektoniske rammeverk kan nivåene i Create illustreres som figur 4.4. Figur 4.4 har et ekstra nivå i form av et sikkerhetslag som beskrevet over.


**Figur 4.4:** Arkitektoniske nivå i Create.

## 4.3 Utviklingsprosess

### 4.3.1 Læringsfasiliteter

Ettersom vi ikke hadde noen formell opplæring i Create ble utviklingen av applikasjonen og opplæring i lavkodeplattformen en og samme prosess. Appfarm ga oss tilgang til tre vesentlige læringsfasiliteter – dokumentasjon, Showroom og Slack. I dokumentasjonen finnes grunnleggende informasjon om Appfarm og Create. Dokumentasjonen består av en introduksjon som gir et overblikk over nøkkelkonsepter, hvordan en applikasjon utvikles, testes og distribueres. I dokumentasjonen er det en Referanseguide med innføring til strukturen i Create. Her finnes det eksempler som illustrerer brukergrensesnitt og hvordan handlingsnodene fungerer. Det er også en seksjon med ulike artikler om emner som ytelsesoptimering og sikkerhetskonfigurasjon. For å komme i gang med opplæringen benyttet vi oss av seksjonen kalt Appcademy som består av en rekke kortere opplæringsvideoer delt i to segmenter. Det første segmentet varer i til sammen 55 minutter og består av en generell introduksjon med navigasjon og nøkkelkonsepter i Create. Neste segment varer i til sammen 45 minutter og demonstrerer utviklingen av et grensesnitt for en applikasjon med enkel oppgavebehandling. Her er det mulig å følge videoene eller utføre de samme aktivitetene selv i etterkant.

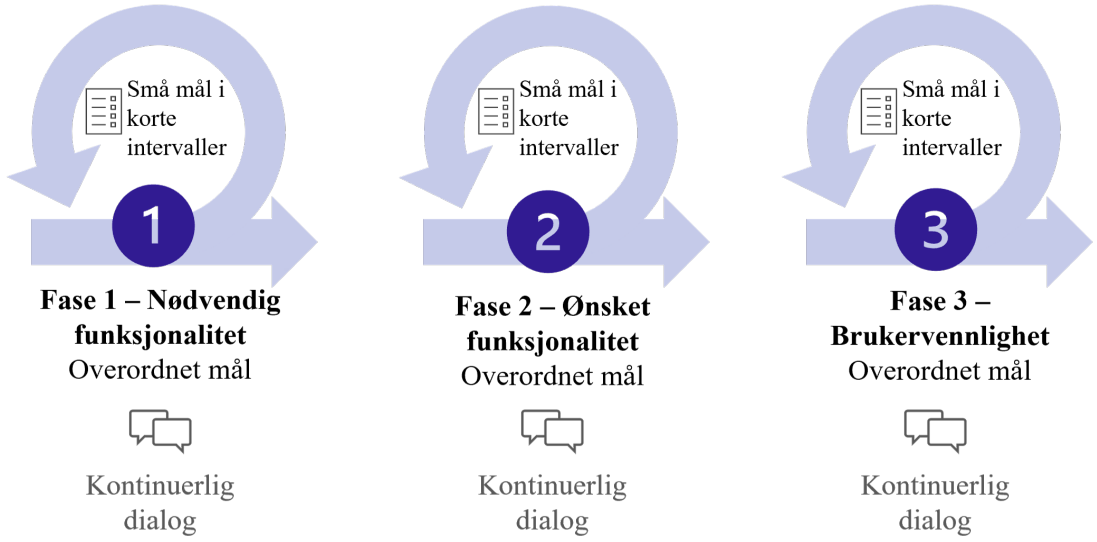
*Showroom* er en løsning i Create som inneholder maler og eksempler. Dette var spesielt nyttig å få tilgang til da det illustrer et spekter av muligheter og logikk-oppbygging. Her kan man for eksempel sjekke hvordan handlingsnoder, datalagring og APIer er satt opp, i tillegg kan man teste funksjonaliteten i forhåndsvisningen av applikasjonen. Det var en god inspirasjonskilde da vi var usikre på hvordan vi skulle løse en utfordring underveis i utviklingsprosessen.

Vi fikk tilgang til diskusjonskanalene til Appfarm i Slack. Alle som bruker utviklingsplattformen, kan her stille spørsmål hvor både ansatte i Appfarm og andre utviklere kan komme med forslag og drøfte mulige løsninger på problemer som oppstår i utviklingsprosessen. Søkefunksjonen gjorde at vi kunne finne frem til spørsmål og svar som var gitt tidligere, derfor har vi også brukt Slack som et oppslagsverk når vi møtte utfordringer underveis.

### 4.3.2 Strategi for utviklingsprosessen

For å forstå et kodespråk eller kodegenerering i en programvare har vi erfart at det er viktig å starte enkelt og bygge på funksjonalitet og kompleksitet etter hvert som forståelsen vår forbedres. Vi delte utviklingsarbeidet opp i tre faser – nødvendig funksjonalitet, ønsket funksjonalitet og brukervennlighet. Siden vi ikke hadde kunnskap og erfaring i Create fra før, var det uforutsigbart hvor lang tid vi kom til å bruke på ulike delene av utviklingsprosessen. Det var derfor lite hensiktsmessig å sette andre tidsfrister enn at applikasjonen må ferdigstilles i samsvar med progresjonen for masteroppgaven. Designvitenskapelig forskningsmetode er som nevnt bygget på iterasjoner og evalueringer underveis i metoden. Dette praktiseres i utviklingsprosessen.

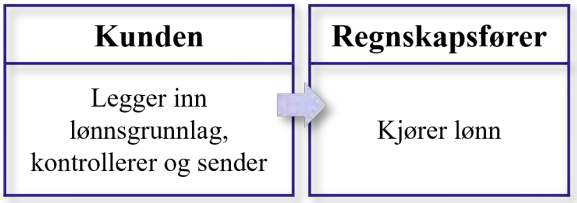
Innen prosjektledelse i IT-prosjekter og utvikling er det populært å bruke rammeverk som for eksempel Scrum og Kanban for agile utviklingsprosesser. Dette er rammeverk som bygger på samarbeid, fremgang og forbedringer, spesielt utviklet for teamarbeid i arbeidslivet. I disse rammeverkene er også iterasjoner sentralt. Vi benytter elementer fra de to rammeverkene. De tre fasene kan sammenlignes med sprints i Scrum-rammeverket. Vi hadde kontinuerlig dialog og skrev ned små mål med korte tidsintervaller for hver fase. Dette for å bidra til progresjon og fleksibilitet i utviklingen. På denne måten hadde vi til enhver tid oversikt over hvor vi skulle, hva som var under arbeid og hvor langt vi var kommet. Utviklingsstrategien er illustrert i figur 4.5 og den praktiske tilnærmingen er beskrevet videre i dette kapittelet.




**Figur 4.5:** Utviklingsstrategi.

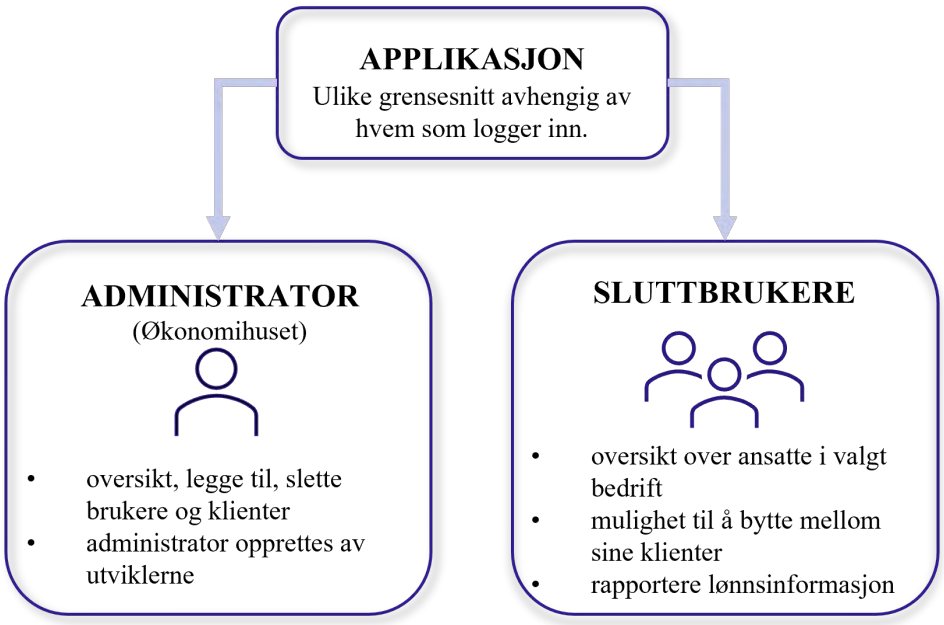
### 4.3.3 Nødvendig funksjonalitet

Før vi startet arbeidet i Create måtte vi diskutere og komme frem til ønsket prosess ved bruk av applikasjonen. Målet var å fjerne regnskapsførers manuelle arbeid og dermed redusere risiko for menneskelige feil. Vi kom frem til følgende prosess ved bruk applikasjonen:




**Figur 4.6:** Illustrasjon av ny prosess.

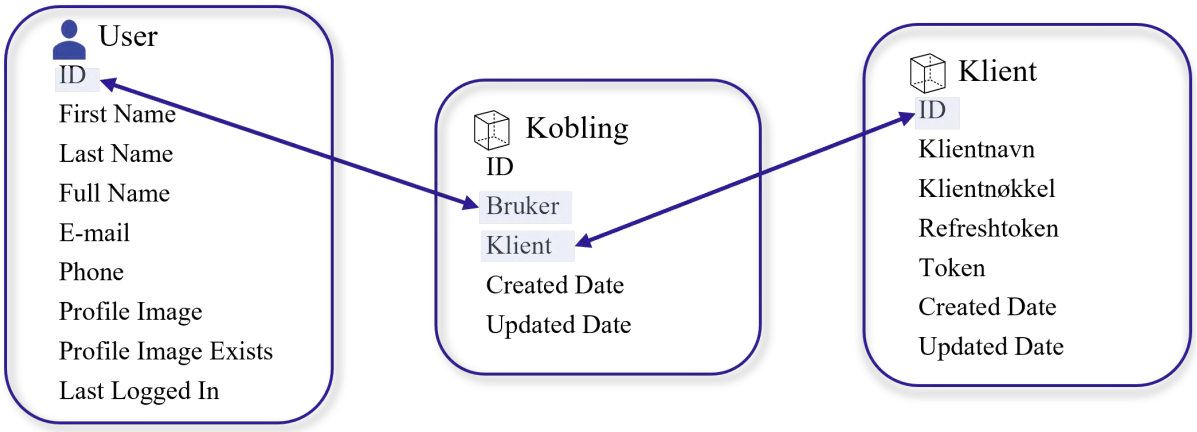
Deretter skisserte vi hvordan applikasjonen burde se ut og noterte hvilke funksjonaliteter den burde ha. Vi tok utgangspunkt i at applikasjonen trengte to ulike grensesnitt avhengig av hvem som logger inn i applikasjonen. Her skiller vi mellom administrator og bruker. Administrator vil være en ansatt i Økonomihuset som har ansvar for tilganger og klienter i applikasjonen. Vi kaller Økonomihuset sine kunder, altså bedriftene som benytter seg av regnskapstjenestene, for klienter fordi dette samsvarer med terminologien som brukes i GO. En bruker er en person som skal legge inn lønnsgrunnlag for klienten. Dette kan for eksempel være daglig leder (bruker) i bedriften (klient).




**Figur 4.7:** Fase 1 - Nødvendig funksjonalitet.

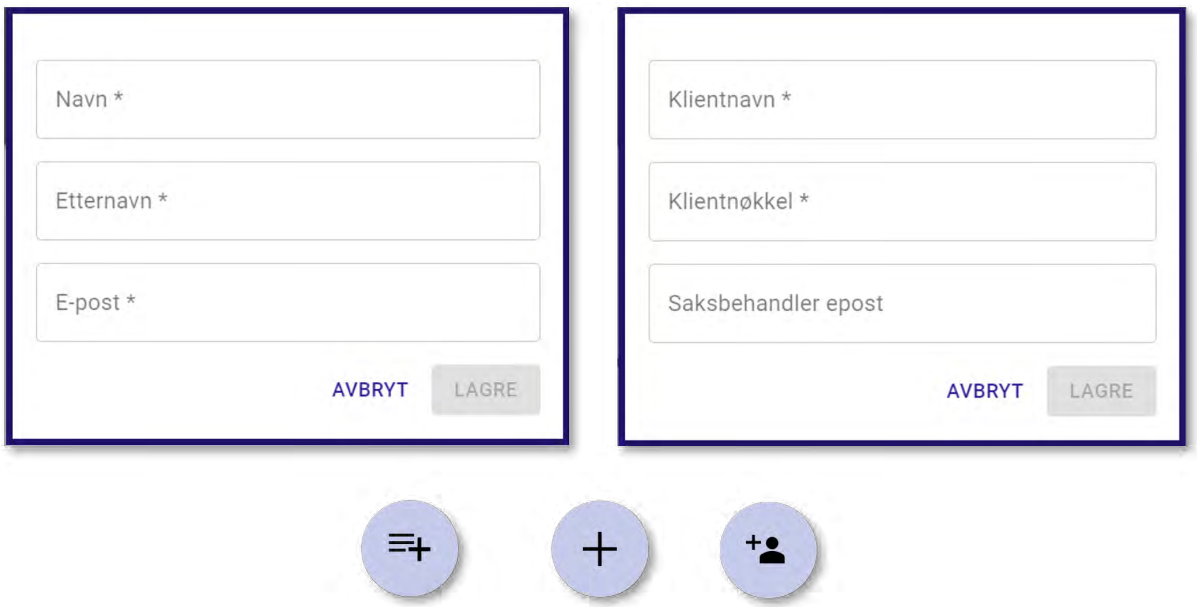
#### 4.3.3.1 Brukergrensesnitt for administrator

Første steg i utviklingsprosessen ble dermed å lage et funksjonelt grensesnitt for en administrator. Vi startet med å sette opp en datamodell for å håndtere brukere og klienter. I tillegg måtte det være mulig å administrere koblingen mellom disse, med andre ord hvilke brukere som skal ha tilgang til hvilke klienter. Koblingen settes opp som en mange-til-mange kobling med en egen objektklasse hvor hvert objekt er knyttet til en bruker og en klient gjennom objektenes ID.



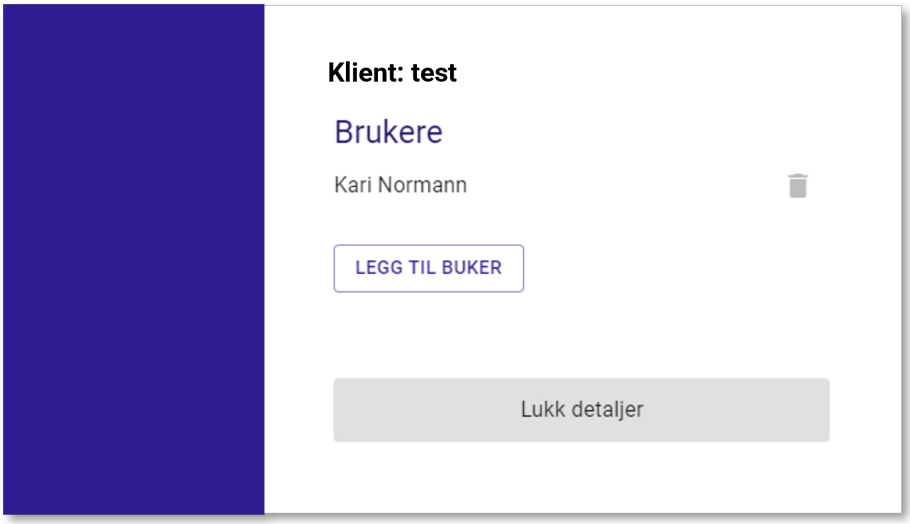

**Figur 4.8:** Datamodell for brukere, klienter og kobling.

I Create er det ulike byggeklosser som kan brukes til å lage brukergrensesnittet. Siden fase 1 fokuserer på funksjonalitet fremfor design, benyttet vi de eksisterende byggeklossene. Vi la først til tre knapper, en for hver funksjon. Knappene for å legge til bruker, legge til klient og legge til kobling åpner alle en dialogboks hvor administrator kan legge inn nødvendig informasjon. Feltene i dialogboksene er knyttet til en egenskap i det aktuelle objektet.




**Figur 4.9:** Dialogbokser og funksjonsknapper for administrator.

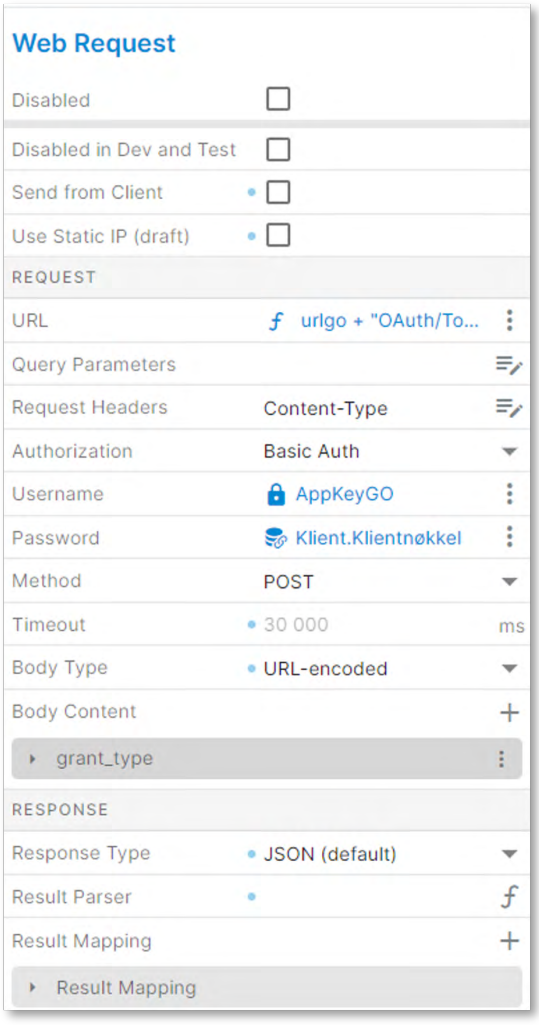
Vi oppdaget fort at det ikke var hensiktsmessig å administrere koblingene på denne måten. Derfor valgte vi heller å vise en liste over alle eksisterende klienter med mulighet for redigering. Vi la til to handlingsknapper på hver klient, den ene gir administrator muligheten til å slette klienten, mens den andre åpner en dialogboks med detaljer. I dialogboksen får administrator opp en liste med brukere som er tilknyttet klienten. Her kan administrator slette brukertilganger og legge til nye og på denne måten opprette koblingene. Knappene for å opprette klienter og brukere beholdt vi.




**Figur 4.10:** Dialogboks med liste over brukere som er tilknyttet klienten.

#### 4.3.3.2 API-tilkobling

Når vi hadde et funksjonelt grensesnitt for administrator kunne vi gå videre til sluttbruker. Her møtte vi to utfordringer, den ene var å kontrollere tilgangen slik at brukeren kun får opp de klientene hen har tilgang til. Den andre var tilkoblingen til APIet til GO. GO sitt API benytter en protokoll kalt *oAuth2* til autentisering for tilkobling. Hver applikasjon har en unik applikasjonsøkkel på tvers av alle klienter. I tillegg har hver klient en klientnøkkel per applikasjon. For å sette opp en ny klient må administrator hente klientnøkkelen fra GO og legge den inn i vår applikasjon når klienten opprettes. Disse nøklene brukes til å hente en tilgangstoken som er gyldig i 10 minutter om gangen. Denne tokenen må følge alle forespørsler om henting eller sending av informasjon. I Create finnes det en egen handlingsnode for HTTP(S)-forespørsler som skal muliggjøre tilkobling til APIer uten koding. Figur 4.11 viser oppsettet for å hente tilgangstoken for en klient.




**Figur 4.11:** Oppsett i Create for å hente tilgangstoken for en klient i GO.

Appfarm anbefaler i sin dokumentasjon å teste forespørslene i verktøyet Postman som er en plattform for å bygge og bruke APIer. Plattformen inneholder en rekke verktøy for å designe, teste og dokumentere APIer (Postman u.å.). Vi forsøkte først å sette opp spørringen i Create men opplevde det som tungvint å teste og feilsøke i løsningen. Derfor valgte vi å følge Appfarm sin anbefaling og testet videre i Postman hvor vi enkelt kunne se responstekst og tilpasse spørringen. Da vi hadde en fungerende forespørsel i Postman la vi denne inn i Create og fikk til en gyldig forespørsel med tilgangstoken i retur. Neste steg var å hente informasjon fra GO om klientens ansatte og lagre disse. For å få det til måtte først objektet *ansatte* opprettes i datamodellen med nødvendige egenskaper. Vi trengte navn, ansattnummer og ID fra GO, i tillegg til en kobling til klienten den ansatte tilhører. For å kunne registrere endringer, men unngå dobbel lagring, benyttet vi en løkkefunskjon som oppdaterer alle ansatte og legger til nye hver gang disse lastes inn fra GO.

Da informasjon skulle sendes tilbake til GO fikk vi problemer med oppsettet. Informasjonen som sendes til GO for postering i systemet skal sendes i JSON. Vi antok at dette var mulig å gjøre uten manuell koding og forsøkte de ulike formatene i Create. Dette fungerte ikke og det endte tilslutt med at vi måtte skrive teksten selv i JavaScript som vist i figuren under.

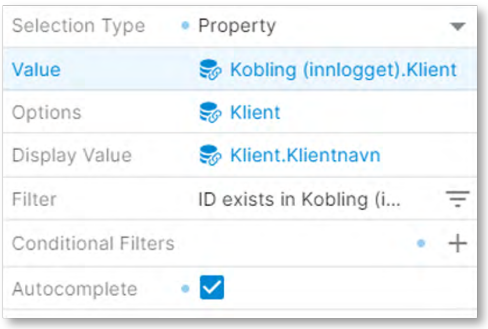
```
if (beskrivelse == undefined)
        return '{"employeeCode":"' + nummer +'","payItemCode":"'+ kode +
                    '","quantity":"'+ antall + '","rate":"' + sats +
                    '","amount":"' + belop + '"}'

else
        return '{"employeeCode":"' + nummer +'","payItemCode":"'+ kode +
                    '","quantity":"'+ antall + '","rate":"' + sats +
                    '","amount":"' + belop + '","comment":"' + beskrivelse +'"}'
```

**Figur 4.12:** JavaScript for informasjon som sendes til GO.

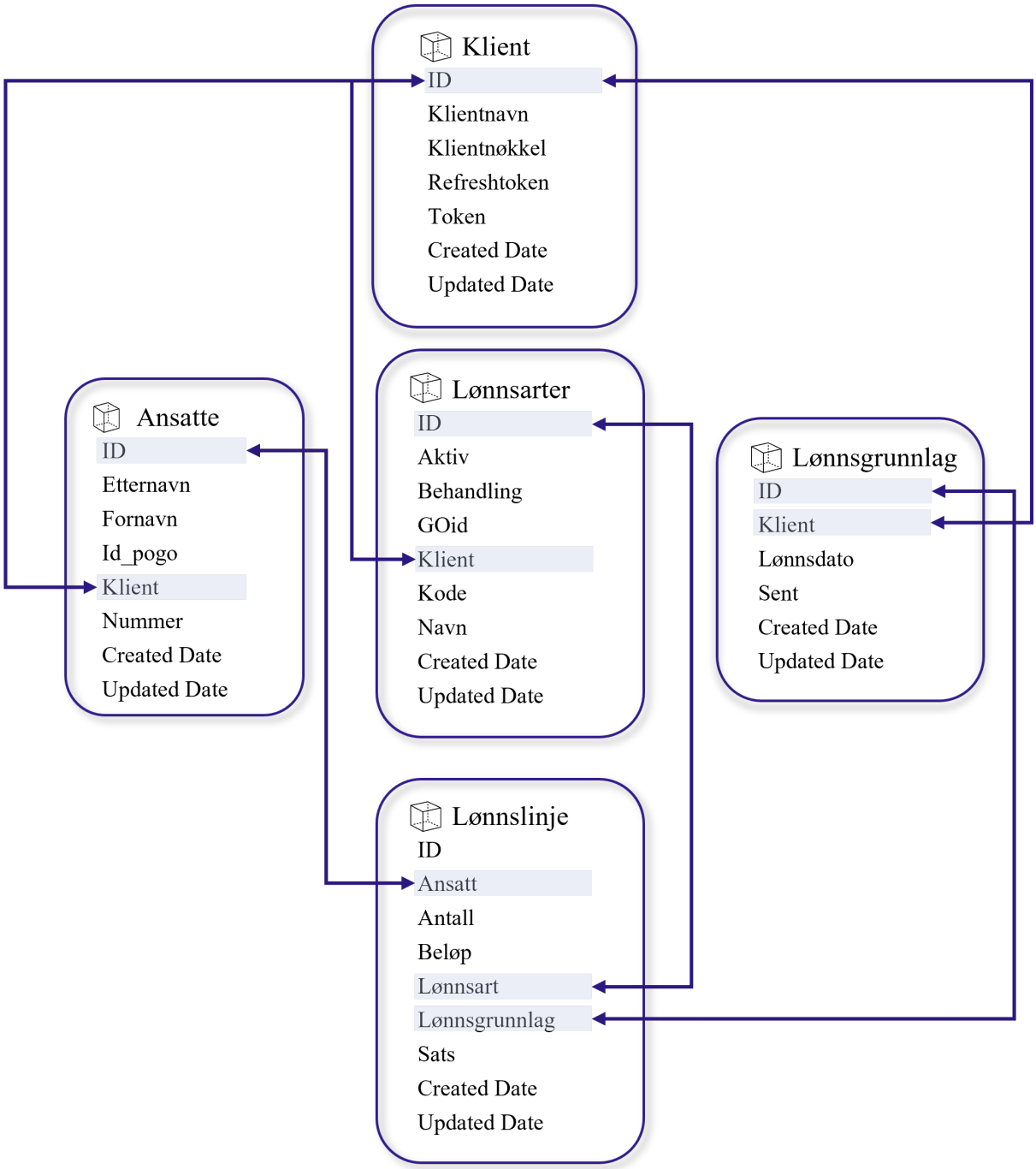
#### 4.3.3.3 Sluttbrukergrensesnitt

For sluttbrukere ønsket vi et enkelt oppsett hvor bruker kan sende inn lønnsinformasjon på de klientene hen har tilgang til. Vi tok utgangspunkt i en liste med ansatte med mulighet for å legge til lønnslinjer på hver ansatt. Videre måtte vi sikre at brukeren bare har tilgang til de klientene som er definert av administrator og at det er mulig å navigere mellom disse. For å styre tilgangen bruker vi et filtrert objekt med alle klientene som eksisterte i brukerens koblinger. Oppsettet er vist i figuren under.



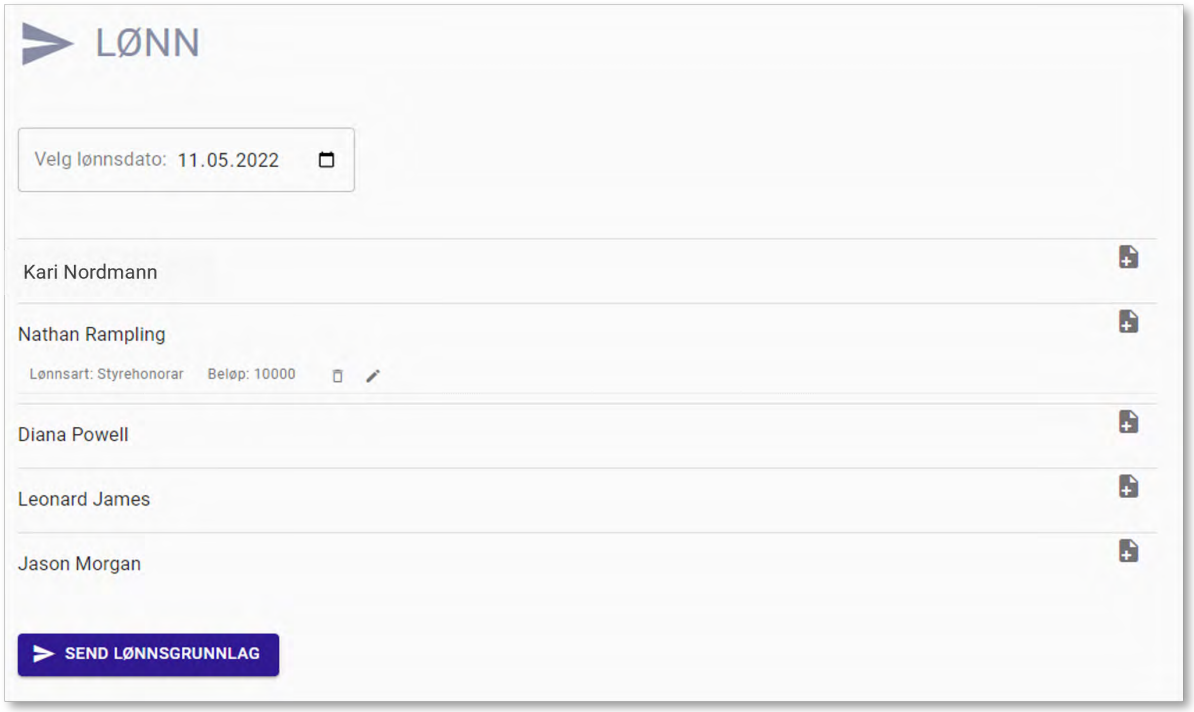

**Figur 4.13:** Oppsett i Create for å filtrere koblingen mellom klienter og brukerens tilganger.

Neste steg var datamodellen for lønnsgrunnlagene. Et lønnsgrunnlag kan bestå av flere lønnslinjer og må være knyttet til riktig klient med tilhørende lønnsarter og ansatte. I denne datamodellen benyttes det mange-til-en forhold mellom de ulike objektene. En klient kan ha flere ansatte, lønnsarter og lønnsgrunnlag, men hvert av de objektene kan bare være knyttet til en klient. Videre kan hvert lønnsgrunnlag ha mange lønnslinjer, men hver lønnslinje kan bare være knyttet til ett lønnsgrunnlag. En grafisk oppstilling av denne datamodellen vises i figur 4.14.



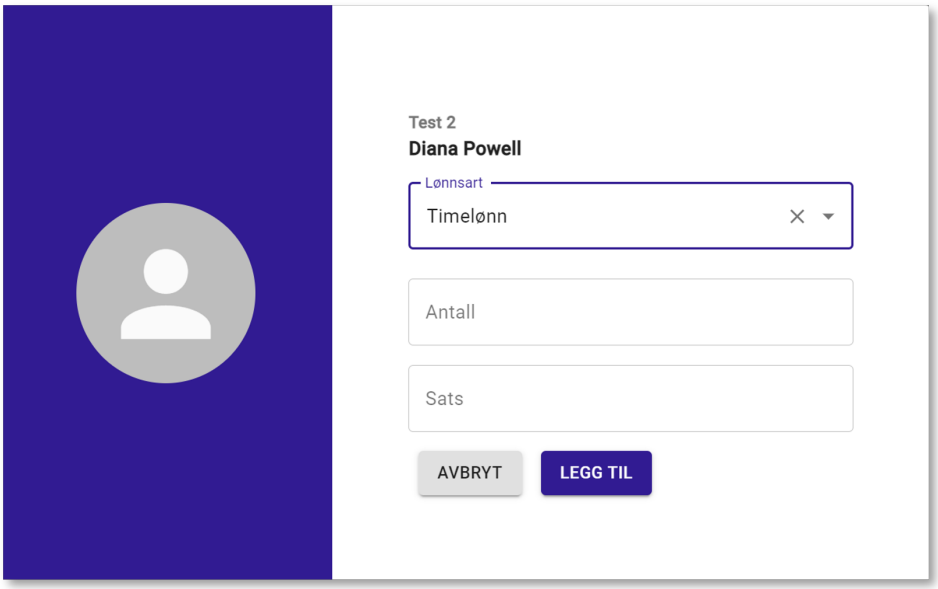

**Figur 4.14:** Datamodell for lønnsgrunnlaget.

Da datamodellen var på plass kunne vi designe brukergrensesnittet. Vi laget en liste med de ansatte, en datovelger og en knapp for å bekrefte og sende inn lønnsgrunnlaget.



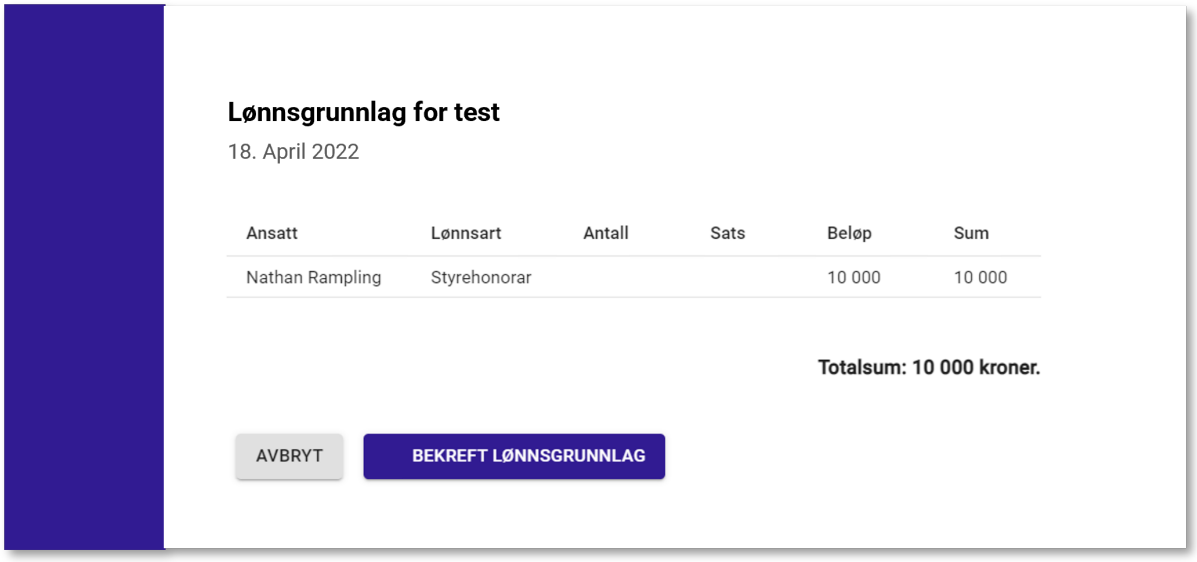

**Figur 4.15:** Illustrasjon av brukergrensesnitt for å legge til lønnsgrunnlag.

Ved å trykke på en ansatt får brukeren opp en dialogboks, som vist i figur 4.16 hvor hen kan legge til en ny lønnslinje ved å velge lønnsart og fylle ut antall og sats eller beløp, avhengig av behandlingsmåten til lønnsarten.




**Figur 4.16:** Dialogboks for å legge til en ny lønnslinje.

Før lønnsgrunnlaget blir sendt til GO ønsket vi å vise brukeren en oppsummering som et ekstra sikkerhetsledd. Oppsummeringen kommer i en ny dialogboks hvor brukeren må bekrefte lønnskjøringen før den sendes til GO.




**Figur 4.17:** Dialogboks med oppsummering og bekreftelse av lønnsgrunnlag.

Ved slutten av fase 1 i utviklingsprosessen har vi en funksjonell applikasjon hvor administrator kan styre brukertilganger og klienter. Sluttbruker kan opprette og sende inn et lønnsgrunnlag.

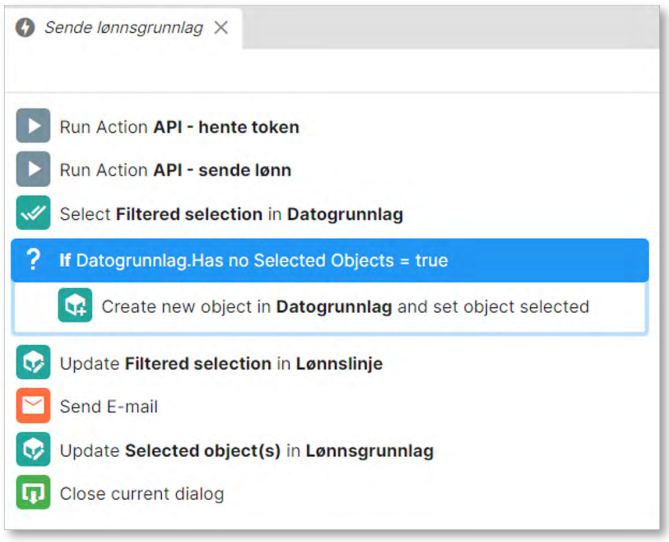
### 4.3.4 Ønsket funksjonalitet

Før vi startet neste runde med utvikling viste vi applikasjonen til representanter fra Økonomihuset for en evaluering og diskusjon om ønsket funksjonalitet. På bakgrunn av diskusjonen laget vi følgende prioriteringsliste:

1. arkiv med mulighet for å kopiere lønnsgrunnlag
2. informasjon om hvem som har sendt inn lønnsgrunnlaget
3. import av timesats på ansatte
4. import av lønnsarter på klientnivå

Vi begynte med å utvikle arkivet som en liste med lønnsgrunnlag presentert som datoer. Sluttbruker kan sende inn flere lønnsgrunnlag med samme dato, disse lagres som sperate

grunnlag i datamodellen. Vi kom derfor frem til at det kunne være nyttig å slå sammen lønnsgrunnlag for samme dato. Først forsøkte vi å gjøre dette med den opprinnelige datamodellen vår, noe som viste seg å være vanskelig. Vi landet etter hvert på en ny objektklasse i datamodellen kalt *Datogrunnlag*. Her opprettes det et ny oppføring ved innsending av lønnsgrunnlag dersom det ikke allerede eksisterer en oppføring på samme dato og klient. Deretter knyttes alle lønnslinjene til denne oppføringen. Med denne strukturen på plass kunne vi enkelt sette opp et arkiv hvor brukeren kan klikke seg inn på detaljene for hver dato.




**Figur 4.18:** Handlingsnode for å sende lønnsgrunnlag.

Kopieringsfunksjonen var på dette stadiet enkel å legge til. Lønnslinjene i de aktuelle grunnlagene blir kopiert inn til et åpent grunnlag og brukeren sendes dit slik at hen kan gjøre eventuelle endringer før innsending.

Økonomihuset mente løsningen burde inneholde informasjon om hvem som har levert grunnlaget av dokumentasjons- og sikkerhetsgrunner. Create har støtte for å identifisere brukeren og det eneste vi trengte å gjøre var dermed å legge til en egenskap på objektet *Lønnsgrunnlag* og knytte denne til brukeren.

Videre ønsket vi å importere timesatsen på de ansatte dersom det er lagt inn i GO. Dette viste seg å være vanskelig, da vi ikke fant informasjonen i API-dokumentasjonen til GO. Vi sendte en mail til Poweroffice og de bekreftet at det ikke er mulig å hente ut lønnsrelatert informasjon via API.

GO har et sett av standard lønnsarter som kan redigeres av regnskapsfører. For å sikre at lønnsinformasjon sendes med riktig lønnsart må derfor lønnsartene hentes fra GO for den enkelte klient. Opprinnelig tenkte vi at lønnsartene kunne lastes inn automatisk ved oppstart av applikasjonen slik at de til en hver tid samsvarte med GO. Da vi satt opp denne funksjonen så vi fort at det var upraktisk at brukeren hadde tilgang til alle lønnsartene som finnes. En mulig løsning var å deaktivere de lønnsartene som ikke er i bruk i GO og bruke et filter på spørringen slik at kun aktive lønnsarter lastes inn. Dette ble også upraktisk da det er tidkrevende å endre alle lønnsartene i GO, i tillegg til at det innførte en feilkilde utenfor administrators kontroll. Derfor bestemte vi oss for å legge lønnsartene inn i administratorløsningen, slik at administrator laster de inn og huker av for hvilke lønnsarter som skal være aktive for klienten. Innlastingen skjer automatisk ved oppretting av en ny klient, og deretter kan administrator trigge ny import ved behov. Fra samtaler med Økonomihuset vet vi at det sjeldent gjøres endringer i lønnsarter etter oppstart av en klient, og det er derfor tilstrekkelig at administrator har tilgang til denne funksjonen.

### 4.3.5 Design og brukervennlighet

En vesentlig forutsetning for at applikasjonen skal lykkes er at den er det foretrukne valget for sluttbrukerne. I tillegg må den være enkel å bruke for administrator slik at de ansatte i Økonomihuset også opplever løsningen som tidsbesparende. Hittil i utviklingen har vi fokusert på at applikasjonen skal være funksjonell og at handlingsnodene og integrasjonene fungerer som de skal. Siden sluttbruker antas å være minst teknisk og mest skeptisk til nye løsninger startet vi forbedringen med kundeflaten. Det viktigste aspektet i applikasjonen er oppretting og innsending av lønnsgrunnlaget. Vi fikk ansatte i Økonomihuset til å teste løsningen for å gi oss et bedre grunnlag for forbedringer. Vi fikk fort tilbakemelding om at oppretting av lønnsgrunnlag krevde for mange tastetrykk og fremsto rotete og uoversiktlig. En ansatt i Økonomihuset viste oss et regneark som hadde vært brukt til innhenting av grunnlag til det tidligere regnskapssystemet deres. Han mente at en tabell som kundene kan navigere i med tastaturet vil være enklere og mer effektivt. På bakgrunn av dette bestemte vi oss for å forsøke å lage en slik tabell. Create har en byggekloss som er en tabell, men denne tilfredsstilte ikke våre ønsker til utforming og funksjon. Vi bygget derfor opp tabellen fra bunnen ved hjelp av repeterende beholdere og byggeklosser for

tekstredigering.

Etter at siden for lønnsgrunnlag var bygget på nytt gjorde vi mindre designendringer på resterende sider. Vi flyttet blant annet klientvelgeren opp i toppbanneren og samkjørte alle dialogboksene og designet mellom administrator- og sluttbrukergrensesnitt. Vi vurderte i tillegg fargevalg og plassering av knapper for å lede brukeren naturlig gjennom applikasjonen. For sluttbruker har vi for eksempel laget kryss for sletting av lønnslinje i rødt og knapp for ny lønnslinje i grønt. Dette for å hjelpe sluttbruker å forstå funksjonaliteten. I siste gjennomgang bestemte vi oss for at det også kan være nyttig med et sammendrag på ansattnivå. Vi ønsket at denne skulle komme fortløpende og valgte derfor å legge den på høyre siden av skjermen.

## 4.4 Produkt

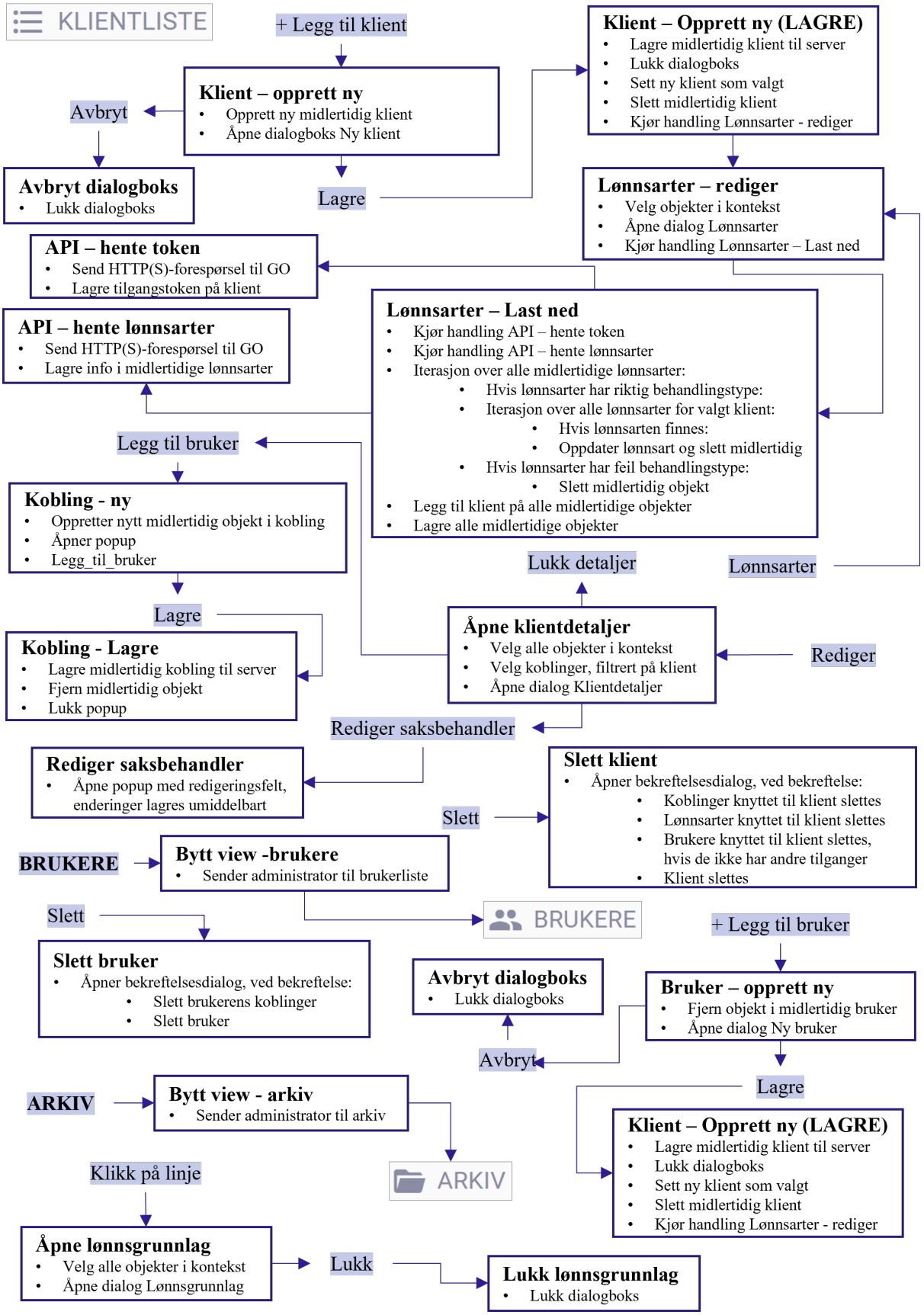
Sluttproduktet er en todelt applikasjon med ett grensesnitt for administrator og ett for sluttbrukere. Brukeren sendes til den mest relevante siden. For administrator er startsiden klientlisten og for sluttbruker er det bildet for innsending av lønnsgrunnlag. Under beskrives funksjonaliteten i applikasjonen kort. Hva som skjer bak det visuelle er illustrert i figur 4.19 og 4.20.

### 4.4.1 Administratorgrensesnitt

Administrator har tilgang til tre faner; Klienter, Arkiv og Brukere. I klientlisten kan administrator legge til nye og gjøre endringer på eksisterende klienter. For å legge til en ny klient benyttes en plussknapp på toppen av tabellen. Denne knappen åpner en dialogboks hvor klientnavn, klientnøkkel fra GO og eventuell saksbehandler legges inn. Ved å flytte musen over klientlisten blir to knapper synlig på den aktuelle raden. En for redigering av klienten og en for lønnsarter. Ved å trykke på knappen for redigering kan administrator legge til eller fjerne tilganger til klienten. Det er også mulig å slette klienter. Når administrator trykker på sletteknappen åpnes en bekreftelsesdialog slik at det ikke skal være mulig å slette en klient ved et uhell. Ved sletting av en klient slettes også alle tilhørende tilganger og brukere dersom de ikke har tilgang til andre klienter.

I arkivet kan administrator se alle innsendte lønnsgrunnlag. Det er mulig å sortere, filtrere og søke i tabellen.

I brukeroversikten legger administrator til nye brukere av løsningen. Her kan også brukere slettes. Samme bekreftelse som i klientoversikten vises og dersom en bruker slettes blir også alle tilhørende koblinger til klienter slettet. Figur 4.19 viser hvordan applikasjonen for administratorgrensesnittet fungerer i sin helhet. For bilder av hvordan applikasjonen ser ut for administrator henvises det til appendiks A2 Applikasjon: Administratorgrensesnitt.



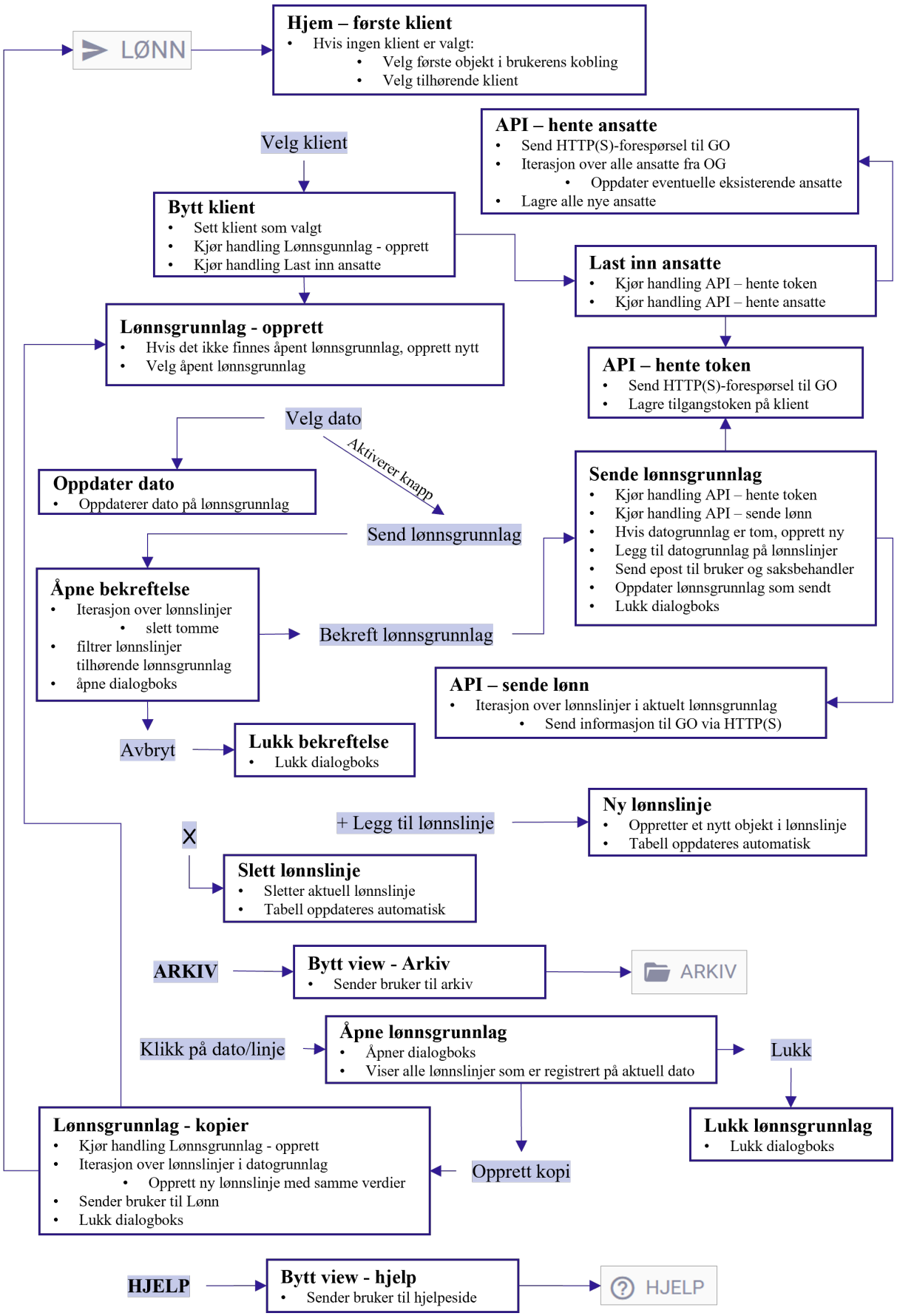

**Figur 4.19:** En oversikt over funksjonene i administratorgrensesnitt.

### 4.4.2 Sluttbrukergrensesnitt

Sluttbruker har også tilgang til tre faner: Lønn, Arkiv og Hjelp. Når sluttbruker logger inn i applikasjonen kommer hen til siden for Lønn og automatisk til siste klient som er opprettet for brukeren. Øverst i venstre hjørne kan brukeren bytte mellom klienter ved bruk av en nedtrekksmeny. For å kunne sende et lønnsgrunnlag må sluttbrukeren velge dato i datovelgeren og legge til lønnslinjer ved å trykke på "Legg til lønnslinje" under tabellen. Sistnevnte knapp legger til en rad som må fylles ut. I første kolonne finner brukeren en nedtrekksmeny som viser klientens ansatte. Neste kolonne er en nedtrekksmeny med lønnsarter. I påfølgende kolonner må sluttbruker fylle inn timesats og antall timer eller beløp avhengig av lønnsart. Det er også mulig å legge til en kommentar ved behov. En lønnslinje kan enkelt slettes ved å krysse ut raden. Etter hvert som lønnslinjer fylles ut oppdateres et sammendrag med sum per ansatt og totalsum for lønnsgrunnlaget til høyre for tabellen. Når hele lønnsgrunnlaget for gjeldene lønnskjøring er fylt ut kan sluttbruker trykke på "Send lønnsgrunnlag". Da dukker det opp en dialogboks med et sammendrag for lønnskjøringen som kan bekreftes eller avbrytes av brukeren. Når brukeren bekrefter lønnsgrunnlaget sendes informasjonen til GO.

Arkivet består av en liste med innsendte lønnsgrunnlag. Listen er sortert på dato. Sluttbruker kan både søke i og filtrere listen. Ved å trykke på datoraden vises en dialogboks med alle lønnslinjene som tidligere er sendt inn for den aktuelle datoen. Brukeren kan enten lukke dialogboksen eller kopiere alle lønnslinjene. Trykker brukeren på "Opprett kopi" lastes siden Lønn hvor lønnslinjene fra kopien er fylt inn automatisk. Brukeren kan gjøre endringer om nødvendig, før lønnsgrunnlaget eventuelt bekreftes og sendes til GO.

Hjelpefanen inneholder instruksjoner og illustrasjoner som beskriver hvordan applikasjonen brukes. Dette for å hindre forespørsler om brukerstøtte til saksbehandler og Økonomihuset. Figur 4.20 viser en fullstendig oversikt over hvordan applikasjonen for sluttbrukergrensesnitt fungerer. For bilder av hvordan applikasjonen ser ut for sluttbruker henvises det til appendiks A3 Applikasjon: Sluttbrukergrensesnitt.




**Figur 4.20:** En oversikt over funksjonene i sluttbrukergrensesnitt.

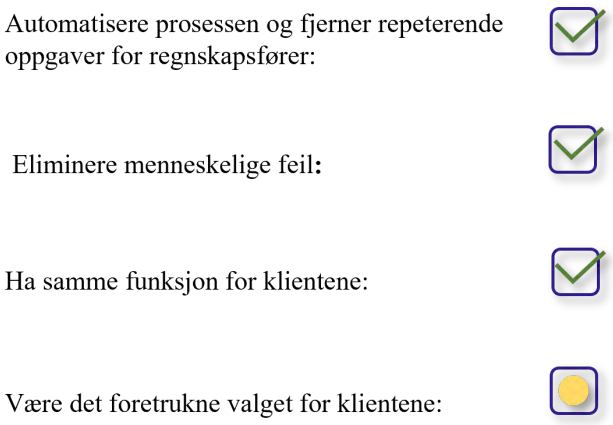
## 4.5 Evaluering

Dette delkapittelet er delt i en evaluering av artefakten og en evaluering av lavkodeverktøy som konstruksjonsprosess. Først redegjør vi for artefakten spesielt, altså hvorvidt eksemplaret forbedrer prosessen for innhenting av lønnsgrunnlag. I utviklingsarbeid snakkes det gjerne om *beste praksis*. Som et ledd av evalueringen presenterer vi derfor forbedringer tilført applikasjonen etter en gjennomgang med Appfarm. Til slutt presenterer vi vårt inntrykk av lavkodeplattformen som konstruksjonsprosess målt opp mot kriteriene som er satt i metodekapittelet.

### 4.5.1 Evaluering av artefakt

Resultatet av applikasjonsutviklingen regnes i hovedsak som vellykket, da prosessen ved bruk av applikasjonen samsvarer med den beskrevet i figur 4.6. Det gjenstår likevel en brukertest med sluttbruker for å avgjøre om løsningen er foretrukket fremfor den eksisterende prosessen. Løsningen er kun testet av ansatte i Økonomihuset. De er forøvrig tilfreds med løsningen og ser at den vil effektivisere arbeidet på deres side. Ettersom de også har vært involvert underveis i utviklingen har vi fått implementert og tilpasset elementer de regnet som viktige for kontroll og funksjon. Eksempler på dette er identifisering av brukeren som har levert grunnlaget og e-postvarsling til saksbehandler.

Som det fremgår av figur 4.21 er alle kriteriene tilknyttet artefakten vellykket foruten punktet om å være det fortrukne valget. Vi har markert denne som gul fordi applikasjonen effektiviserer klientene prosess ved å eliminere en dobbel kontrollrunde fra deres side. Til tross for at vi ikke har hatt tid til å gjennomføre brukerundersøkelser hos Økonomihuset sine klienter har vi fokusert på å designe applikasjonen tilnærmet lik slik klientene ellers ville ha skrevet en e-post. Vi vurderer det derfor som sannsynlig at ny prosess er fortrukket fremfor den eksisterende.




**Figur 4.21:** Kriterier knyttet til prosessforbedring.

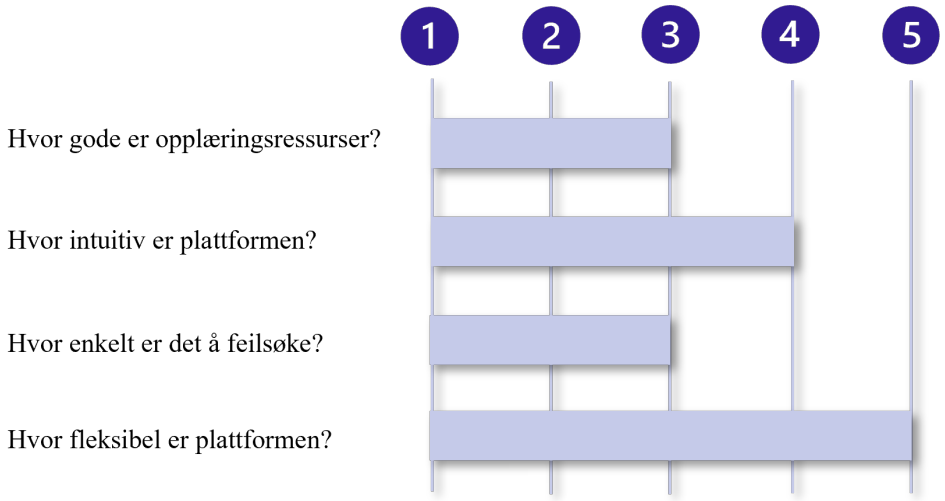
Etter ferdigstilling presenterte vi løsningen for Appfarm. De var enige i at vi hadde laget en funksjonell applikasjon, men ønsket å ta en grundigere gjennomgang av beste praksis. I gjennomgangen ble det fokusert på ytelsesoptimalisering. Vi så nærmere på to ting som kunne skade prestasjonen til applikasjonen. Appfarm behandler datamodeller når løsningen kjøres i brukerens nettleser. Dersom det ikke legges filter på objektklassene i applikasjonsdata lastes alle objekter inn. Dette er hensiktsmessig dersom brukeren har nytte av alle objektene. I flere tilfeller hadde vi satt opp logikken slik at alle objekter lastes inn med et filter for å vise relevante oppføringer for brukeren. Dette opptar unødvendig kapasitet og kan svekke ytelsen. Vi endret derfor vår applikasjon slik at den samsvarer med beste praksis. Det andre potensielle problemet var håndtering av lønnslinjer. Når en lønnslinje opprettes lagres denne direkte i datamodellen istedenfor i midlertidige kjøretidsobjekter. Dette var en delvis bevisst løsning fra vår side, da vi ønsket at brukeren skulle ha mulighet til å starte på et grunnlag og fullføre det senere. Vi diskuterte problemstillingen med Appfarm og kom frem til at det kunne være aktuelt å bruke kjøretidsobjekt og en knapp for lagring, men vi ble enig om å beholde det originale oppsettet på grunn av brukervennlighet. Det er også lite sannsynlig at applikasjonen har mange brukere som arbeider på akkurat samme tid og ytelsen knyttet til spørringene mot server vil derfor ikke være et stort problem.

Selv om Økonomihuset opplever applikasjonen som en forbedring av dagens prosess er omfanget av disse kundene ganske lite. Det er dermed begreneset hvor stort potensialet for besparelser er ved eliminering av det manuelle arbeidet. Kostnaden ved å bruke

applikasjonen er høy ved få brukere, og med de prisene Appfarm har per nå er startprisen 15 000 kr. per måned. Denne prisen inkluderer 150 brukere. Økonomihuset har nå omtrent 30 klienter som er aktuelle for løsningen. Saksbehandlere bruker i snitt 10 minutter per kunde på å legge inn og kontrollere lønnsgrunnlaget. Økonomihuset kalkulerer at dette koster 145 kr., totalkostnaden for en måned blir 4 350 kr. Omfanget er derfor for lite til at bruk av applikasjonen blir lønnsomt. Økonomihuset vil undersøke hvorvidt andre regnskapsbyråer kan være interessert i å bruke applikasjonen. Dersom det finnes stor nok interesse for løsningen hos andre byråer kan det være aktuelt å lisensiere den ut for å dekke kostnaden.

### 4.5.2 Evaluering av konstruksjonsprosessen

Gjennom utviklingen av artefakten har vi gjort oss egne erfaringer rundt bruken av lavkodeverktøyet Appfarm Create. Prosessen har vært en kombinert lærings- og utviklingsprosess hvor prøving og feiling har vært nødvendig. Vi ser at at den typiske utviklingsprosessen presentert av Sahay et al. (2020) også er representativ for vår utviklingsprosess. Figur 4.22 viser vår vurdering av kriteriene satt i kapittel 3.2.




**Figur 4.22:** Vurdering av Appfarm Create.

Introduksjonen og opplæringen til Appfarm oppleves uferdig og delvis utdatert da en del elementer har endret både navn og plassering i Create. Dette gjorde at vi ikke hadde et fullstendig bilde av funksjonalitet da vi startet utviklingen. Appfram er klar over

utfordringene med læringsressursene og holder i disse dager på å ferdigstille en oppdatert versjon. Det er derfor mulig at introduksjonen vil oppleves enklere for kommende brukere. Dokumentasjonen og ressursene som var tilgjengelig for oppslag underveis i utviklingen er likevel gode og vi gir derfor læringsressursene en samlet vurdering midt på treet.

Logikken i Create er stort sett lett forståelig og intuitiv, men vi savnet noen funksjoner underveis. Muligheten for å angre slik vi kjenner fra de fleste dataprogrammer var det største savnet. Det var litt for lett å slette et element ved et uhell, og var en uheldig og slettet en beholder med mye innhold ble det tungvindt å bygge denne på nytt. Videre savnet vi hurtigtaster for blant annet kopiering. En annen ting vi opplevde som frustrende var at en hake for "les alle objekter" må krysses av dersom det ikke brukes filter på en objektklasse som skal inkluderes i datamodellen. Dersom dette ikke er huket av leses ikke objektene i løsningen. Etter optimaliseringspraten med Appfarm forstår vi imidlertid logikken bak dette, da Appfarm i hovedsak legger opp til bruk av filtrert data. Dette er et eksempel på informasjon som kommer for dårlig frem i opplæringsressursene.

Verktøyet for HTTP[S]-forespørsler kunne vært mer intuitivt og vi mener det burde inneholdt mulighet for automatisk formatering i JSON. Det kan være krevende å bruke JavaScript til å generere tekst i JSON når det skal inkluderes variabler som ligger i datamodellen. En enklere løsning for dette vil senke brukerterskelen. Vi savnet også enklere muligheter for å regne med variabler, da det må gjøres i JavaScript. Noen av de ferdige elementene manglet designmuligheter som gjorde at vi valgte å lage spesialtilpassede elementer. Dette gjelder blant annet formateringsmuligheter i tabeller og mulighet for inntasting i datovelgeren. Basert på disse erfaringene mener vi Appfarm har et stykke igjen før de når visjonen om å være en nullkodeplattform. Dette samsvarer også med informasjonen vi fikk fra representanten fra Avo i forkant av utviklingen. Vi velger likevel å rangere plattformen i sin helhet som relativt intuitiv da de elementene som oppleves som ferdigutviklet er gode og enkle å jobbe med.

Feilsøking i applikasjonen fungerer stort sett godt gjennom feilsøkingsverktøyet. Denne ligger nederst på siden når det kjøres en forhåndsvisning i utviklingsmiljøet. Her står det hvilke handlinger som er kjørt og om de feilet. God informasjon om hvorfor handlingen feilet mangler i noen tilfeller, men siden det pekes på riktig handling klarte vi å løse alle problemene som oppstod. Feilsøking av komplekse enkeltelementer som for eksempel

JavaScript og HTTP[S]-forespørsler kunne vært gjort enklere. På disse områdene savnet vi en mulighet for å teste og feilsøke i utviklingen uten å kjøre forhåndsvisning av applikasjonen. På bakgrunn av sistnevnte rangeres feilsøking i løsningen som verken god eller dårlig.

Til tross for enkelte utfordringer underveis har vi lykkes med å utvikle en fungerende applikasjon med minimalt av teknisk bistand underveis. Utviklingen har gått relativt fort, og vi har stort sett kost oss med arbeidet og opplevd mestering underveis. Det er sjeldent vi har stått fast ved en problemstilling over lengre tid, og vi har ikke møtt problemer som ikke har latt seg løse. Vår applikasjon er ikke veldig kompleks, men gjennom vår erfaring har vi ikke noe å utsette på fleksibiliteten i verktøyet og vurderer derfor denne som svært god.

Vår samlede erfaring er at Appfarm er et intuitivt og nyttig verktøy som kan brukes av personer med teknisk interesse men uten formell kompetanse. Plattformen har fortsatt stort forbedringspotensiale på enkelte punker og en utbedring av disse vil kunne senke brukerterskelen ytterligere.

# 5 Diskusjon

Det stilles stadig høyere krav til å utnytte digitale verktøy for å effektivisere leveringen av regnskapstjenestene. Automatisk dataoverføring er essensielt for å sikre effektivitet i prosesser, og med stor grad av skybaserte systemer er API-integrasjoner relevant. I denne oppgaven undersøker vi om regnskapsførerselskap kan benytte lavkodeverktøy til å automatisere arbeidsoppgaver og dermed bidra til den digitale utviklingen. For å besvare forskningsspørsmålet diskuterer vi først bruken av lavkodeverktøy generelt, deretter anvendelse i regnskapsbransjen og til slutt et økonomisk perspektiv.

## 5.1 Nytteverdi og brukervennlighet

Det fremkommer av evalueringen at vi synes Create er et intuitivt og nyttig verktøy som passer godt for personer som er teknisk interesserte. Dette er en subjektiv betrakting av ett enkelt lavkodeverktøy. Det er klart at det med manglende standarder i markedet, uten noe felles rammeverk eller læringsfasiliteter vil være stor variasjon på tvers av tilbyderne. Vi kan derfor ikke trekke konklusjoner på hvorvidt lavkodeverktøy generelt er enkelt å forholde seg til med vår bakgrunn. Vi ser likevel at tjenestene til Appfarm samsvarer med typisk arkitektur, infrastruktur og utviklingsprosess presentert av Sahay et al. (2020) i kapittel 2.3.3. Gitt at lavkodeplattformen er intuitiv for brukeren er det en rekke faktorer som taler for å benytte seg av teknologien.

Den største fordelen med lavkodeplattformer er en kortere utviklingsprosess (se for eksempel Bock og Frank, 2021; Sanchis et al., 2020; Waszkowski, 2019). Gjennom arbeidet med utviklingen av artefakten finner vi støtte for dette. På kort tid har vi lært oss bruke Create og utviklet en velfungerende applikasjon. Med kompetansen vi har i Create i dag mener vi at det er mulig å utvikle enkle, fungerende løsninger på kun noen få dager.

Forskning fremmer fordelene med skybaserte utviklingsplattformer (Di Ruscio et al., 2022). Det pekes på lavere inngangsbarrierer og oppstartskostnader. Dette ble også tydelig gjennom arbeidet med denne masteroppgaven. Oppstarten i Create tok kun noen minutter og ingen nedlasting av programvare var nødvendig. Vi gikk løs på utviklingen umiddelbart, parallelt med opplæringen.

Med kort utviklingstid og lave inngangsbarrierer hevdes det at lavkodeplattformer egner seg godt til å teste nye idéer uten for store kostnader (Richardson og Rymer, 2016). Påstanden bekreftes av samtaler med både Avo og Appfarm. De viser til flere tilfeller hvor kundene bruker Create til å utvikle en MVP (Minimum Viable Product). I ett tilfelle ble første versjon av løsningen lansert i markedet allerede etter to uker. Dette samsvarer med våre egne erfaringer, da vi hadde en fungerende løsning som kunne testes av Økonomihuset etter kort tid. Dette gjør det også enkelt å tilpasse applikasjonen og implementere ønskede funksjoner etter lansering. Ønsker man for eksempel en ny sletteknapp kan det ordnes på få minutter. Dette er en faktor som gjør at lavkodeplattformer kan fungere som en innovasjonsarena som kan bidra til effektive løsninger som gjerne ikke hadde blitt realisert dersom det eneste alternativet var tradisjonell koding.

Med begrenset utviklingskompetanse og ingen IT-utdanning i bakhånd, har vi klart å utvikle en funksjonell applikasjon som tilfredsstiller forhåndsbestemte krav og ønskede funksjoner. Dette støtter påstanden om at at lavkodeplattformer muliggjør utvikling for andre enn de med formell IT-utdanning. I Norge er det estimert et behov for 40 000 nye ansatte med IKT-kompetanse innen 2030 (Eggen et al., 2021). Det er særlig mangel på programvare- og applikasjonsutviklere. Kompetansemangelen er et vesentlig hinder for hele den digitale utviklingen (KPMG, 2020). Lavkodeutvikling kan være med å løse dette behovet da plattformene kan benyttes av personer uten programmeringsbakgrunn. I følge Appfarm fjerner en behovet for en rekke utviklingsressurser ved å benytte lavkodeverktøy. Det er for eksempel ikke behov for en sikkerhetsekspert og egne backend og frontend utviklere som ellers ville vært nødvendig i et tradisjonelt utviklingsprosjekt.

Til tross for at lavkode fjerner behovet for enkelte ressurser kreves det fortsatt kompetanse på flere områder for å utvikle en vellykket løsning. I vår utviklingsprosess hadde vi en egen fase som fokuserte på brukervennlighet. Interaksjonsdesign er en forskningsbasert spesialisering innen informatikk, i form av UX designere (User Experience). I samtalene med Avo fremkommer det at selskapet investerer i ytterligere UX kompetanse for å sikre brukervennlighet ved digitale tjenester. Dette er et viktig aspekt. Et kriterium for vårt eksemplar var at den nye prosessen skulle være det foretrukne valget for kundene til Økonomihuset og design og brukervennlighet er derfor avgjørende. For å dra nytte av lavkodeverktøy kreves det mer enn bare teknisk interesse, det er også behov for

interesse og kunnskap om design. Dersom lavkodeverktøy skal være en del av for eksempel regnskapsførers kompetanse er det en risiko for at kunnskapen blir overfladisk. Mangel på spesialisering kan ha motsatt effekt og resultere i dårlige og lite effektive produkter.

## 5.2 Lavkodeverktøy i regnskapsbransjen

Regnskapsbransjen beveger seg vekk fra manuelle og repeterende oppgaver. I relevant litteratur presenteres fire drivkrefter for en digital transformasjon i regnskapsbransjen (Prem, 2015). Lavkodeverktøy tilrettelegger for å ta i bruk disse driverne, noe vi illustrerer med vår applikasjon. Digital innsamling og prosessering er løst med et internettgrensesnitt mot kunden. Videre er applikasjonen integrert med regnskapssystemet og informasjonen overføres i sanntid og sikrer automatisert verdiskapning. Utfordringen som er løst i denne masteroppgaven demonstrerer at lavkodeverktøy kan brukes til å utvikle løsninger som er nyttige for regnskapsbransjen.

Økonomihuset hadde flere problemstillinger som ble vurdert som aktuelle for en lavkodeløsning. Sammen med Økonomihuset identifiserte vi en rekke interne behov hvor vi mener at lavkode kan effektivisere prosessene. Det er en utfordring for Økonomihuset å finne et regnskapssystem som løser et hvert behov på en effektiv måte. Derfor er det rom for å benytte tilleggssystemer som et supplement. Et eksempel på dette er interne systemer for håndtering av boligselskap, herunder eierskifter, fakturering og tredjepartsopplysninger. Per i dag er dette prosesser som gjennomføres ved bruk av delvis automatiserte Excel-løsninger i kombinasjon med andre programmer. En applikasjon kunne for eksempel inneholdt alle eiere knyttet til de ulike andelene i hvert boligselskap. Videre kunne applikasjonen vært integrert med GO slik at eierskifter blir fakturert og ny eier blir satt opp til å motta månedlig faktura på felleskostnader. Til slutt kunne applikasjonen vært integrert med Altinn. Sistnevnte slik at tredjepartsopplysningene som baserer seg på eiertid og kostnader, inntekter, formue og gjeld i det aktuelle året kan rapporteres automatisk.

Det kan være en fordel å inkludere ansatte i utviklingsprosessen innen egne fagfelt (Di Ruscio et al., 2022; Metrôlho et al., 2020). Det er strenge krav og lover knyttet til regnskapsføring i Norge. Regnskapsfaget har, som andre fag, egne terminologier og logikk som ikke nødvendigvis utenforstående kjenner til. Det kan derfor være effektivt å la fagpersoner fra bransjen stå for utviklingen av teknologien som skal brukes i samspill

med regnskapet. Regnskapsfører bruker allerede en rekke digitale verktøy i sin jobb. Som det kommer frem i kapittel 2.1 finnes det få regnskapsførere uten systemkompetanse, da det er et krav for å håndtere oppgavene effektivt. Fra vårt arbeid med masteroppgaven ser vi at det er mulig for personer med teknisk interesse, men uten formell kompetanse, å utvikle fungerene løsninger ved hjelp av lavkode. Det er derfor sannsynlig at enkelte regnskapsførere er i stand til å bruke lavkodeverktøy. Økonomihuset mener selv at omtrent 15 prosent av de ansatte vil kunne bidra i slikt arbeid basert på eksisterende kunskap og interesse.

En annen faktor som kommer frem i samtaler med Økonomihuset er at regnskapsførere har et behov for å forstå teknologien for å stole på at den fungerer tilfredsstillende. Ledere i Økonomihuset sier at dette behovet kommer tydelig frem i flere tilfeller, for eksempel ved bruk av automatiske funksjoner i regnskapssystemet. Et eksempel på en slik funksjon er automatisk bankavstemming og bokføring av enkelte transaksjoner. Denne funksjonen er foreløpig lite i bruk, og Økonomihuset mener at det blant annet skyldes manglende forståelse for teknologien og uro for å miste kontroll over regnskapet. Involvering i utviklingsprosessen kan muligens være med å sikre forståelse for det ferdige produktet. Dette kan imidlertid også være et hinder for anvendelse av lavkode i bransjen. Det er trolig bare de mest tekniske regnskapsførerene som vil ønske å involvere seg i utviklingsarbeidet. Dersom man ønsker å ta i bruk lavkode er det dermed svært viktig å dokumentere arbeidet og sørge for at brukerinstruksene er gode. Videre bør det finnes mulighet for å kontrollere resultatet applikasjonene produserer, for eksempel gjennom loggføring.

Tidligere studier foreslår en rolleendring i retning av en mer rådgivende funksjon (for eksempel Schei et al., 2015). I tillegg legges det vekt på at regnskapsfører må utvikle sin kompetanse i takt med den digitale utviklingen (Marrone og Hazelton, 2019). Etter arbeidet med lavkodeløsningen mener vi at det kan være fornuftig at regnskapsførerselskapene styrker sin utviklingskompetanse. Dette kan sette dem i stand til å identifisere muligheter som skapes av ny teknologi og i noen tilfeller også gjennomføre utviklingsarbeidet. Et slikt kompetanseløft kan gjøre at regnskapsførerselskapene stiller sterkere mot potensielle nye konkurrenter fra andre yrkesgrupper. Det kan diskuteres hvorvidt selskapene bør rekruttere nye ansatte med en mer teknisk bakgrunn. Samtidig er det viktig å beholde fagkompetansen. Forskning peker på et behov for å tilpasse utdanningsløpet til regnskapsfører slik at det er

mer relevant for arbeidsoppgavene i bransjen fremover (Al-Htaybat et al., 2018). Dersom teknisk kompetanse får plass i utdanningen kan det bidra til å skape mer relevante arbeidstakere som også kan bidra i digitaliseringsarbeidet i bedriftene. I en digital verden vil dette, sammen med fagkompetanse, gi regnskapsførerrollen mer legitimitet.

Digitaliseringsgraden i regnskapsførerselskap påvirkes av digitaliseringsgraden hos sluttkundene (Baksaas et al., 2019). Regnskapsfører sitter ofte tett på kunden og ser problemstillinger fra både et regnskapsperspektiv og fra kundens bedriftsperspektiv. I denne posisjonen kan det tenkes at regnskapsfører vil være godt egnet til å skreddersy løsninger til sine kunder. Regnskapsførerselskap som har mulighet til å hjelpe kundene med digitaliseringsarbeidet vil også øke digitaliseringsgraden innad i eget selskap.

## 5.3 Økonomisk perspektiv

Lavkodeplattformer presenterer en mulighet for kostnadsreduksjon i utviklingsprosjekter. Tidsaspektet er vesentlig når det kommer til vurdering av kostnader knyttet til en ny løsning. Ved bruk av lavkodeverktøy kan løsningen som nevnt raskt utvikles og testes i markedet. Dette reduserer risiko i prosjektet og gir mulighet for å forkaste mislykkede løsninger uten store kostnader. Alternativet, dersom man er avhengig av spesialutviklede verktøy, vil ofte være dyrere. Tradisjonell utvikling tar lengre tid og krever mer kompetanse og flere ressurser.

Regnskapsbransjen beveger seg fra timepris over mot fastpris og verdibasert prising (Regnskap Norge, u.å.). Dette gjør at en eventuell effektivisering av interne rutiner vil komme regnskapsbyrået til gode. Lavkodeplattformer kan være kostbare og det kreves derfor et visst volum før løsningene er lønnsomme. Løpende kostnader knyttet til drift av lavkodeløsninger inneholder flere elementer, blant annet lisenskostnader og kostnader knyttet til vedlikehold og videreutvikling. Vi erfarer at lisenskostnaden kan være en barriere for bruk. Dette er særlig tilfellet der omfanget av løsningen er lite, som i vårt eksempel dersom løsningen bare tas i bruk internt i Økonomihuset.

Å være i stand til å utvikle løsninger for kunder kan sikre regnskapsbyrået nye inntektskilder når regnskapsoppgavene er mer automatisert. Ansatte i Økonomihuset opplever det som vanskelig å overbevise mindre kunder om at ny, kostbar teknologi er nødvendig. Det er derfor mest aktuelt å utvikle løsninger for kunder av en viss størrelse, eller et visst

volum av mindre kunder som kan dele på investeringen. Det kan også være aktuelt for regnskapsførerselskapet å bære kostnaden dersom løsningen også effektiviserer deres arbeid.

Systemkostnader er en vesentlig kostnadsdriver for regnskapsbyråer. En undersøkelse fra Regnskap Norge beskriver systemene som generelt dyre og prisene som uoversiktlige. De påpeker også at skybaserte løsninger og andre fremskritt gjør det billigere å programmere selv (Kvalheim, 2022). Dersom regnskapsførerselskapene kan ta del i utviklingen av egne systemer kan de redusere bruken av eksterne løsninger og kostnadene knyttet til disse.

# 6 Konklusjon

Denne oppgaven søker å besvare følgende forskningsspørsmål:

> *"Er lavkodeverktøy egnet til å automatisere arbeidsoppgaver i regnskapsbransjen og bidra til regnskapsførerselskapenes digitale utvikling?"*

Lavkodeverktøy kan brukes til å automatisere prosesser i regnskapsførerselskap, dette har vi demonstrert gjennom et konkret eksempel. I tillegg har vi diskutert hvordan lavkodeverktøy kan brukes til å effektivisere andre arbeidsoppgaver. Vår applikasjon sikrer digital datahåndtering og automatisert verdiskapning gjennom integrasjoner og digitale brukergrensesnitt og viser dermed at lavkodeverktøy kan bidra til den digitale utviklingen i regnskapsførerselskap. Oppgaven viser at personer med begreneset teknisk kompetanse kan være i stand til å benytte lavkodeplattformer.

Vi diskuterer også lavkodeverktøy fra et økonomisk perspektiv. Effektivisering av repeterende og manuelle arbeidsprosesser kan medføre en større besparelse enn kostnaden ved utvikling og drift av en applikasjon. Siden bransjen beveger seg over til fastpris vil en reduksjon i antall timer ha direkte effekt på bunnlinjen. Økt grad av digitalisering krever stadig flere lisensierte systemer som er en vesentlig kostnadsdriver for bransjen. Ved bruk av lavkode kan regnskapsførerselskapene i større grad styre sitt eget digitaliseringsarbeid og redusere innflytelsen fra de store systemleverandørene. Digitaliseringsgraden i regnskapsbyråene er avhengig av digitaliseringsgraden til kundene. Ved å utvikle løsninger for kunder kan dermed byråene bli mer digitalisert samtidig som de sikrer nye inntektskilder.

Våre funn gir grunnlag for å konkludere med at lavkodeverktøy kan brukes i regnskapsbransjen for å automatisere enkelte arbeidsprosesser og bidra til den digitale utviklingen i regnskapsførerselskapene.

## 6.1 Kunnskapsbidrag

I denne oppgaven vurderer vi en relativt ny teknologi som er lite forsket på. Lavkodekonseptet markedsføres som brukervennlig og med lave krav til forkunnskaper, vi finner likevel lite forskning som støtter denne påstanden. Studien vår undersøker funksjonaliteten til et lavkodeverktøy og vurderer hvorvidt dette er egnet for automatisering

av arbeidsprosesser i regnskapsførerselskap.

Studien bidrar med innsikt i hvordan regnskapsførerselskapene kan fortsette å være relevant i den digitale tidsalder. Vår utredning bygger videre på det faktum at bransjen er i endring og kommer med forslag til hvordan regnskapsførerselskapene kan ta en aktiv rolle i den digitale utviklingen. Regnskapslitteraturen fremhever et behov for å kommunisere forskningen på en mer anvendbar måte (Guthrie og Parker, 2016, 2017). Gjennom grundig dokumentasjon av utviklingsarbeidet for en reell utfording forsøker vi å legge til rette for at kunnskapen kan brukes av bransjen i praksis.

Mangelen på applikasjon- og programvareutviklere i Norge gjør en evaluering av lavkodeverktøy særlig relevant. Det er sannsynlig at flere bransjer kan ta i bruk lavkodeteknologi. Innsikten vi kommuniserer gjennom oppgaven kan være nyttig for de virksomhetene som vurderer å bruke lavkodeverktøy. Studien gir også andre som senere ønsker å evaluere tilsvarende plattformer et godt utgangspunkt for sammenlikning.

## 6.2 Begrensinger og forslag til videre forskning

Rammene for masteroppgaven setter begrensninger på bredden av oppgaven og hvor mange aspekter vi kan inkludere. Oppgaven er begrenset til én casebedrift, én lavkodeplattform og én arbeidsprosess. Artefakten vi utvikler i oppgaven vurderes opp mot kriteriene vi satt innledningsvis, men den vurderes ikke i forhold til egnethet for tilpasning og overlevelse i Økonomihuset. Til tross for at vi diskuterer tema mot motiv og utfordringer i litteraturen og bransjen er det mange aspekter som ikke inkluderes i oppgaven. For eksempel diskuterer vi ikke bindingseffekter ved å benytte en lavkodeplattform. For å si noe om dette måtte vi sett nærmere på prosessen med å bytte plattform eller teknologi, noe som ikke er realistisk med den tiden vi hadde til disposisjon. Sikkerhet er et viktig aspekt ved bruk av IT-verktøy. Vi har valgt å se bort i fra sikkerhet i masteroppgaven da Appfarm stiller med sikkerhetsgarantier slik at utviklerne ikke trenger å ta stilling til dette. Mangel på IT-kompetanse kan likvel utgjøre en risiko for sikkerhet og det kan derfor argumenteres for at det bør tas hensyn til sikkerhetsaspektet uavhengig av garantier fra plattformleverandørene.

Vi fremmer eksempler på andre prosesser som kan effektiviseres ved bruk av lavkodeverktøy i Økonomihuset, men innenfor rammene av en masteroppgave er det ikke gjennomførbart

å undersøke bredden av hva verktøyet potensielt kan brukes til. Vi er kort inne på kontrollbehovet som observeres hos regnskapsførere i Økonomihuset, men diskuterer ikke menneskelige forhold utover dette. For eksempel redegjør vi ikke for sosiale konsekvenser som at ansatte generelt er motvillig til endring. Vi sier ingenting om hvordan bruken av lavkodeverktøy bør struktureres og organiseres i virksomheten. Diskusjonen rundt lønnsomhet baserer seg på ett enkelt case og generell informasjon fra bransjeorganisasjonen Regnskap Norge. En nærmere undersøkelse av prismodeller og systemkostnader kunne stryket våre antagelser på området.

Vår forskningsmetode, DSRM, er basert på subjektive refleksjoner og vi har tidligere redegjort for hvordan metoden bidrar til kunnskap. Det er vesentlig med en ytterligere forskningsinnsats for et bredere bilde av hvordan lavkodeverktøy kan utnyttes og tjene regnskapsbransjen. Vi forseslår at videre forskning kan fokusere på hvem som skal besitte lavkodekompetansen og hvorvidt det er hensiktsmessig med tverrfaglige team. Det er en fordel med teori og innsikt i hvordan en investering i lavkodeverktøy bør implementeres i en organisasjon. Det kan også med fordel undersøkes hvordan lavkode bør være en del av en digital strategi i et regnskapsførerselskap. I den forbindelse kan det gjøres en mer omfattende undersøkelse av forventet lønnsomhet. Ettersom lavkodeplattformer hevdes å være heterogene tjenester vil det være fordelaktig med studier som undersøker flere plattformer med fokus på hva som passer bransjen best. Det kan også være interessant å undersøke annen relevant teknologi som et alternativ til lavkode, slik som for eksempel RPA.

# Referanser

Agnew, H. (2016). *Auditing: Pitch battle.* https://www.ft.com/content/268637f6-15c8-11e6-9d98-00386a18e39d.

Agostino, D. & Sidorova, Y. (2017). How social media reshapes action on distant customers: some empirical evidence. *Accounting, Auditing & Accountability Journal.* https://doi.org/10.1108/AAAJ-07-2015-2136.

Al-Htaybat, K., von Alberti-Alhtaybat, L., & Alhatabat, Z. (2018). Educating digital natives for the future: accounting educators' evaluation of the accounting curriculum. *Accounting Education*, 27(4):333–357. https://doi.org/10.1080/09639284.2018.1437758.

Appelbaum, D. & Nehmer, R. (2017). The coming disruption of drones, robots, and bots: how will it affect cpas and accounting practice? *CPA Journal*, 87(6).

Appfarm (2022). *Appfarm Documentation: Key concepts.* Appfarm AS. Hentet 15. april fra lukket kilde.

Arnaboldi, M., Azzone, G., & Sidorova, Y. (2017a). Governing social media: the emergence of hybridised boundary objects. *Accounting, Auditing & Accountability Journal.* https://doi.org/10.1108/AAAJ-07-2015-2132.

Arnaboldi, M., Busco, C., & Cuganesan, S. (2017b). Accounting, accountability, social media and big data: revolution or hype? *Accounting, auditing & accountability journal.* https://doi.org/10.1108/AAAJ-03-2017-2880.

Austheim, S. (20. juni, 2020). *Kartlegging av Markedet for salg av regnskapssystemer.* Regnskap Norge. https://www.regnskapnorge.no/faget/artikler/teknologi2/kartlegging-av-markedet-for-salg-av-regnskapssystemer/.

Baksaas, K. (2019). *Trekk ved god profesjonsutdanning innen regnskap og revisjon*, sider 237–254. Fagbokforlaget.

Baksaas, K. M., Gustavsen, T., Nytun, M., & Stephansen, S. (2019). *Status for digitaliseringen hos regnskapsførers kunder*, sider 73–88. Fagbokforlaget.

Bergvall-Kåreborn, B. & Howcroft, D. (2011). Mobile applications development on apple and google platforms. *Communications of the Association for Information Systems*, 29. https://doi.org/10.17705/1CAIS.02930.

Bock, A. C. & Frank, U. (2021). Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine*, 63(6):733–740. https://doi.org/10.1007/s12599-021-00726-8.

Bygstad, B. (2017). Generative innovation: A comparison of lightweight and heavyweight it. *Journal of Information Technology*, 32(2):180–193. https://doi.org/10.1057/jit.2016.15.

Cabot, J. (2020). *Low-code vs model-driven: are they the same?* Virtual Event, 1-3. https://modeling-languages.com/low-code-vs-model-driven/.

Cleven, A., Gubler, P., & Hüner, K. M. (2009). Design alternatives for the evaluation of design science research artifacts. I *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, sider 1–8. https://doi.org/10.1145/1555619.1555645.

Da Silva, A. R. (2015). Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, 43:139–155. https://doi.org/10.1016/j.cl.2015.06.001.

Datatilsynet (2018). *Skytjenester*. https://www.datatilsynet.no/personvern-pa-ulike-omrader/internett-og-apper/skytjenester/.

Delacruz, E. N. & Lim, J. (2018). *The Nature of Change: Why Companies Need to Adapt or Die*. https://asialeadership.org/2018/09/25/the-nature-of-change-why-companies-need-to-adapt-or-die/.

Di Ruscio, D., Kolovos, D., de Lara, J., Pierantonio, A., Tisi, M., & Wimmer, M. (2022). Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling*, 21:437–446. https://doi.org/10.1007/s10270-021-00970-2.

Eggen, F. W., Måøy, J., Røtnes, R., Nordberg-Schultz, M., & Steen, J. I. (2021). *Norges behov for IKT-kompetanse i dag og fremover - Rapport nr. 1-2021*. Samfunnsøkonomisk analyse AS.

Frey, C. B. & Osborne, M. (2013). The future of employment: How susceptible are jobs to computerisation? *Technological Forecasting and Social Change*. https://doi.org/10.1016/j.techfore.2016.08.019.

Gartner (2021). Gartner says cloud will be the centerpiece of new digital experiences. I *STAMFORD, Conn.* https://www.gartner.com/en/newsroom/press-releases/2021-11-10-gartner-says-cloud-will-be-the-centerpiece-of-new-digital-experiences.

Gartner (u.å.). *Gartner Glossary Digitalization*. Gartner. https://www.gartner.com/en/information-technology/glossary/digitalization.

Geerts, G. L. (2011). A design science research methodology and its application to accounting information systems research. *International Journal of Accounting Information Systems*, 12(2):142–151. Special Issue on Methodologies in AIS Research https://doi.org/10.1016/j.accinf.2011.02.004.

GitHub (25. oktober, 2017). *Opening Keynote - GitHub Universe 2017 [Video]*. YouTube. https://youtu.be/4_umRysuFjU?t=3169.

Gonzalez, R. A. & Sol, H. G. (2012). Validation and design science research in information systems. I *Research methodologies, innovations and philosophies in software systems engineering and information systems*, sider 403–426. IGI Global. https://doi.org/10.4018/978-1-4666-0179-6.ch021.

Granlund, M. & Mouritsen, J. (2003). Special section on management control and new information technologies. *European Accounting Review*, 12(1):77–83. https://doi.org/10.1080/0963818031000087925.

Gustavsen, T. & Baksaas, K. (2019). *Digitalisering - Innhold og trender for regnskapsprofesjonen*, sider 11–27. Fagbokforlaget.

Guthrie, J. & Parker, L. D. (2016). Whither the accounting profession, accountants and accounting researchers? commentary and projections. *Accounting, Auditing & Accountability Journal*. https://doi.org/10.1108/AAAJ-10-2015-2263.

Guthrie, J. & Parker, L. D. (2017). Reflections and projections: 30 years of the interdisciplinary accounting, auditing and accountability search for a fairer society. *Accounting, Auditing & Accountability Journal*. https://doi.org/10.1108/AAAJ-11-2016-2781.

Hanseth, O. & Lyytinen, K. (2010). Design theory for dynamic complexity in information infrastructures: the case of building internet. *Journal of Information Technology*, 25(1):1–19. https://doi.org/10.1057/jit.2009.19.

Hazgui, M. & Gendron, Y. (2015). Blurred roles and elusive boundaries: On contemporary forms of oversight surrounding professional work. *Accounting, auditing & accountability journal*. https://doi.org/10.1108/AAAJ-12-2014-1890.

Heggernes, T. A. (2020). *Digital forretningsforståelse*. Fagbokforlaget.

Helle, L. (5. juni, 2020). *summativ vurdering*. Store Norske Leksikon. https://snl.no/summativ_vurdering.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, sider 75–105. https://doi.org/10.2307/25148625.

Holmen, H. A. (27.desember, 2021). *epistemologi*. Store Norske Leksikon. https://snl.no/epistemologi.

Hubbard, M. (2019). *The Anatomy of an HTTP Request*. https://www.shipengine.com/blog/the-anatomy-of-an-http-request/.

Jensen, M. & Borge-Hansen, P.-Ø. (2020). Digitalisering av økonomifunksjonen. *Praktisk økonomi & finans*, 36(1):10–17. https://www.idunn.no/doi/abs/10.18261/issn.1504-2871-2020-01-03.

Karimi, J. & Walter, Z. (2015). The role of dynamic capabilities in responding to digital disruption: A factor-based study of the newspaper industry. *Journal of Management Information Systems*, 32(1):39–81. https://doi.org/10.1080/07421222.2015.1029380.

Knudsen, D.-R. (2020). Elusive boundaries, power relations, and knowledge production: A systematic review of the literature on digitalization in accounting. *International Journal of Accounting Information Systems*, 36:100441. https://doi.org/10.1016/j.accinf.2019.100441.

Kokina, J. & Davenport, T. H. (2017). The emergence of artificial intelligence: How automation is changing auditing. *Journal of emerging technologies in accounting*, 14(1):115–122. https://doi.org/10.2308/jeta-51730.

Kornberger, M., Pflueger, D., & Mouritsen, J. (2017). Evaluative infrastructures: Accounting for platform organization. *Accounting, Organizations and Society*, 60:79–95. https://doi.org/10.1016/j.aos.2017.05.002.

KPMG (2020). *Hindre for digitalisering av forretningsprosesser. Kommunal- og moderniseringsdepartementet*. KPMG AS. https://www.regjeringen.no/contentassets/cfe173cda0ee461d8aab7ed0eaf5360c/hindringer-for-digitalisering-av-forretningsprosesser.pdf.

Krejci, D., Iho, S., & Missonier, S. (2021). Innovating with employees: an exploratory

study of idea development on low-code development platforms. I *ECIS 2021 Research Papers. 118.* https://aisel.aisnet.org/ecis2021_rp/118/.

Kvalheim, E. (1. juni 2018). Regnskapsføreren før og nå. *Regnskap & Økonomi*. https://www.regnskapnorge.no/magasin/regnskapsforeren---for-og-na/.

Kvalheim, E. (2020). Automatisering gir regnskapsfører nye oppgaver og utfordringer. *Regnskap & Økonomi*. https://www.regnskapnorge.no/magasin/automatisering-gir-regnskapsforer-nye-oppgaver-og-utfordringer/.

Kvalheim, E. (2022). De små aktørene som utfordrer etablerte økonomisystemer. *Regnskap & Økonomi*. https://www.regnskapnorge.no/magasin/de-sma-aktorene-som-utfordrer-etablerte-okonomisystemer/.

Larsen, C. (2021). *Årsmelding 2020*. Regnskap Norge. https://www.regnskapnorge.no/contentassets/91915bbb4fab4ff6a6736d6320866463/rn-aarsmelding-2020.pdf.

Lee, A. S. & Hubona, G. S. (2009). A scientific basis for rigor in information systems research. *MIS quarterly*, sider 237–262. https://doi.org/10.2307/20650291.

Lent, R. W. (2018). Future of work in the digital world: Preparing for instability and opportunity. *The Career Development Quarterly*, 66(3):205–219. https://doi.org/10.1002/cdq.12143.

Lieberman, H., Paternò, F., Klann, M., & Wulf, V. (2006). *End-User Development: An Emerging Paradigm*, sider 1–8. Springer Netherlands, Dordrecht. https://doi.org/10.1007/1-4020-5386-X_1.

Luo, Y., Liang, P., Wang, C., Shahin, M., & Zhan, J. (2021). Characteristics and challenges of low-code development: The practitioners' perspective. I *Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ESEM '21, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3475716.3475782.

March, S. T. & Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems*, 15(4):251–266. https://doi.org/10.1016/0167-9236(94)00041-2.

Marrone, M. & Hazelton, J. (2019). The disruptive and transformative potential of new technologies for accounting, accountants and accountability: A review of current literature and call for further research. *Meditari Accountancy Research*. http://dx.doi.org/10.1108/MEDAR-06-2019-0508.

Martin, J. (1982). *Application Development Without Programmers*. Prentice Hall PTR, Hoboken.

Martin, J. (1991). *Rapid Application Development*. Macmillan Publishing Co. Inc, New York.

Metrôlho, J. C., Ribeiro, F., & Araujo, R. (2020). A strategy for facing new employavility trends using a low-code development platform. I *Conference: 14th International Technology, Education and Development Conference*, sider 8601–8606. http://dx.doi.org/10.21125/inted.2020.2341.

Microsoft (u.å.). *What is SaaS*. https://azure.microsoft.com/nb-no/overview/what-is-saas/.

Miyachi, C. (2018). What is "cloud"? it is time to update the nist definition? *IEEE Cloud Computing*, 5(3):6–11.

Murphy, K. L. (2018). *Robotic Process automation and Low-Code*. https://www.outsystems.com/blog/posts/robotic-process-automation-low-code/.

Næringslivets Hovedorganisasjon (2021). *Økonomisk overblikk 4/2021, Utsikter 2021-2023, Tema: Morgendagens Arbeidsliv*. https://www.nho.no/siteassets/publikasjoner/kvartdalsrapporter/okonomisk-overblikk-4-21.pdf.

Ofoeda, J., Boateng, R., & Effah, J. (2019). Application programming interface (api) research: A review of the past to inform the future. *International Journal of Enterprise Information Systems (IJEIS)*, 15(3):76–95. https://doi.org/10.4018/IJEIS.2019070105.

Osmundsen, K., Iden, J., & Bygstad, B. (2018). Hva er digitalisering, digital innovasjon og digital transformasjon? I *NOKOBIT 2018 - Norsk konferanse for bruk av InformasjonsTeknologi*, volume 26, Svalbard, Norge.

Osmundsen, K., Iden, J., & Bygstad, B. (2019). Organizing robotic process automation: Balancing loose and tight coupling. I *Proceedings of the 52nd Hawaii international conference on system sciences*. http://dx.doi.org/10.24251/HICSS.2019.829.

Parviainen, P., Tihinen, M., Kääriäinen, J., & Teppola, S. (2017). Tackling the digitalization challenge: how to benefit from digitalization in practice. *International journal of information systems and project management*, 5(1):63–77. https://doi.org/10.12821/ijispm050104.

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24:45–77. https://doi.org/10.2753/MIS0742-1222240302.

Postman (u.å.). *Product: Tools*. https://www.postman.com/product/tools/.

Poweroffice (u.å.a). *Api documentation*. https://api.poweroffice.net.

Poweroffice (u.å.b). *Utviderser*. https://poweroffice.no/utvidelser/.

Prem, E. (2015). A digital transformation business model for innovation. I *ISPIM Innovation Symposium*, side 1. The International Society for Professional Innovation Management (ISPIM).

Rainer, K. R. & Prince, P. (2021). *Introduction to information Systems (9. utg.)*. John Wiley and Sons, Inc.

Regnskap Norge (2019). *Bransjerapporten 2018*. Regnskap Norge. https://www.regnskapnorge.no/globalassets/dokumenter/statistikk/bransjerapporten-2018.pdf.

Regnskap Norge (2021a). *Fakta om økonomisystemer*. Regnskap Norge. https://www.regnskapnorge.no/teknologi-og-innovasjon/okonomisystemer_2021/.

Regnskap Norge (2021b). *Teknologiundersøkelsen*. https://www.regnskapnorge.no/teknologi-og-innovasjon/test-av-powerbi/.

Regnskap Norge (u.å.). *Prismodeller*. https://www.regnskapnorge.no/akademiet/kurs/forretningsutvikling2/forretningsutvikling_hjelp/kundedrevet-tjenesteinnovasjon/prismodeller/.

Reis, J., Amorim, M., Melão, N., Cohen, Y., & Rodrigues, M. (2019). Digitalization: A literature review and research agenda. I *International Joint conference on industrial engineering and operations management*, sider 443–456. Springer.

Richardson, C. & Rymer, J. (2014). *New Development Platforms Emerge for Customer-Facing Applications*. Forrester Research, Cambridge.

Richardson, C. & Rymer, J. (2016). *Vendor Landscape: The Fractured, Fertile Terrain Of Low-Code Application Platforms: The Landscape Reflects A Market In Its Formative Years*. Forrester Research, Cambridge.

Riddell, C. (12. januar, 2016). *Digital disruption transforming the finance sector*. Acuity Magazine. https://www.acuitymag.com/opinion/digital-disruption-transforming-the-finance-sector.

Rossen, E. (5. desember, 2019b). *Programmering (IT)*. Store Norske Leksikon. https://snl.no/programmering_-_IT.

Rossen, E. (6.november, 2019a). *IT*. Store Norske Leksikon. https://snl.no/IT.

Rymer, J. (2017). *The Forrester Wave: Low-Code Development Platforms For AD&D Pros, Q4 2017*. Forrester Research, Cambridge.

Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. I *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, sider 171–178. https://doi.org/10.1109/SEAA51224.2020.00036.

Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2020). Low-code as enabler of digital transformation in manufacturing industry. *Applied Sciences*, 10(1). https://www.mdpi.com/2076-3417/10/1/12.

Sandvik, G. (2019). *API – hverdagshelten få kjenner*. https://www.regnskapnorge.no/faget/artikler/teknologi2/api--hverdagshelten-fa-kjenner/.

Schei, V., Sverdrup, T. E., Lund, J., & Riise, S. (2015). Fra transaksjon til relasjon. *Magma*. https://old.magma.no/fra-transaksjon-til-relasjon.

Schmidt, D. C. (2006). Guest editor's introduction: Model-driven engineering. *Computer*, 39:25–31. https://doi.ieeecomputersociety.org/10.1109/MC.2006.58.

Schwab, K. (2017). *The Fourth Industrial Revolution*. Penguin Books Ltd.

Simon, H. (1996). *The Sciences of the Artificial*. MA, MIT Press, Cambridge, third edition edition.

Smith, P. L. (2011). *It i skyen*. Libiris Media A/S.

Spandow, A. (2014). *Fremtidens regnskapsbransje*. https://amestogroup.wordpress.com/2014/03/21/fremtidens-regnskapsbransje/.

Stople, A., Iden, J., Steinsund, H., & Bygstad, B. (2017). Lightweight IT and the IT function: experiences from robotic process automation in a Norwegian bank. I *NOKOBIT, Bibsys Open Journal Systems*, volume 25.

Suddaby, R., Saxton, G. D., & Gunz, S. (2015). Twittering change: The institutional work of domain change in accounting expertise. *Accounting, Organizations and Society*, 45:52–68. https://doi.org/10.1016/j.aos.2015.07.002.

Svartdal, F. (3.april, 2020). *reliabilitet*. Store Norske Leksikon. https://snl.no/reliabilitet.

The Economist (22. august, 2019). *What companies are for*. The Economist Newspaper Limited. https://www.economist.com/leaders/2019/08/22/what-companies-are-for.

Tisi, M., Mottu, J., Kolovos, D., de Lara, J., Guerra, E., Ruscio, D., Pierantonio, A., & Wimmer, M. (2019). Lowcomote: training the next generation of experts in scalable low-code engineering platforms. I *STAF 2019 Co-Located Events Joint Proceedings, volume 2405 of CEUR Workshop Proceedings*, volume 2405, sider 73,78. CEUR-WS.org.

Vaishnavi, V. & Kuechler, W. (2021). Design science research in information systems. Sist oppdatert av forfatterne 24. november 2021. http://www.desrist.org/design-research-in-information-systems/.

Viale, T., Gendron, Y., & Suddaby, R. (2017). From "mad men" to "math men": The rise of expertise in digital measurement and the shaping of online consumer freedom. *Accounting, Auditing & Accountability Journal*. https://doi.org/10.1108/AAAJ-12-2014-1887.

Vincent, P., Natis, Y., Iijima, K., Wong, J., Ray, S., Jain, A., & Leow, A. (2020). *Magic quadrant for enterprise low-code application platforms.* Gartner Report September 2020, Gartner.

Waszkowski, R. (2019). Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine*, 52(10):376–381. https://doi.org/10.1016/j.ifacol.2019.10.060.

Woo, M. (2020). The rise of no/low code software development—no experience needed? *Engineering*, 6(9):960–961. https://doi.org/10.1016/j.eng.2020.07.007.

Yoo, Y. (2010). Computing in everyday life: A call for research on experiential computing. *MIS Quarterly*, 34:213–231. http://doi.org/10.2307/20721425.

# Appendiks

## A1 Innloggingsinformasjon

Applikasjonen er tilgjengelig for test til utgangen av juli 2022. Det er mulig å logge inn i både administratorversjon og sluttbrukerversjon med følgende detaljer:

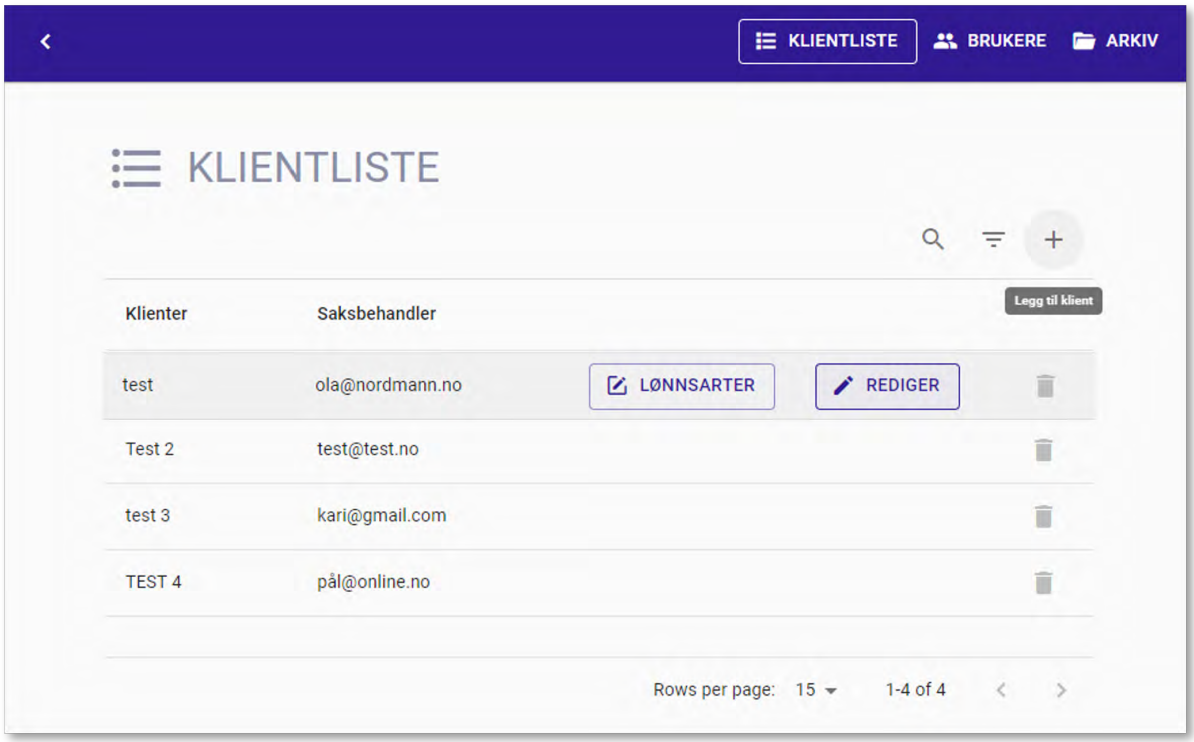
Brukernavn: post@okonomihuset.no
Passord: Testbruker

Trykk her for administratorversjon.
Trykk her for sluttbrukerversjon.

Testbruker har ikke mulighet til å opprette nye brukere. Det er mulig å åpne dialogboksen men lagring vil ikke fungere.

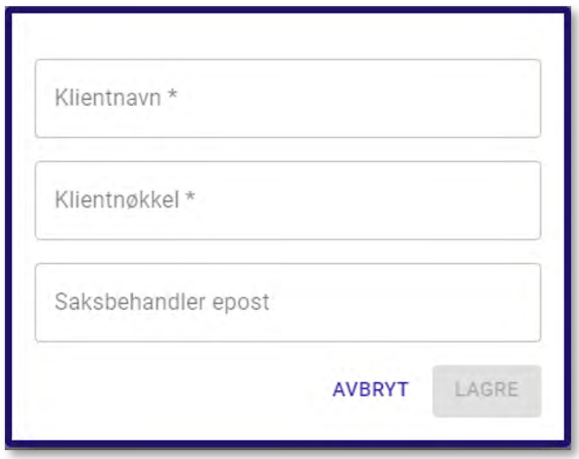
# A2 Applikasjon: Administratorgrensesnitt

Ved innlogging er applikasjonsvisningen på siden Klientliste. Figur A2.1 viser denne siden.



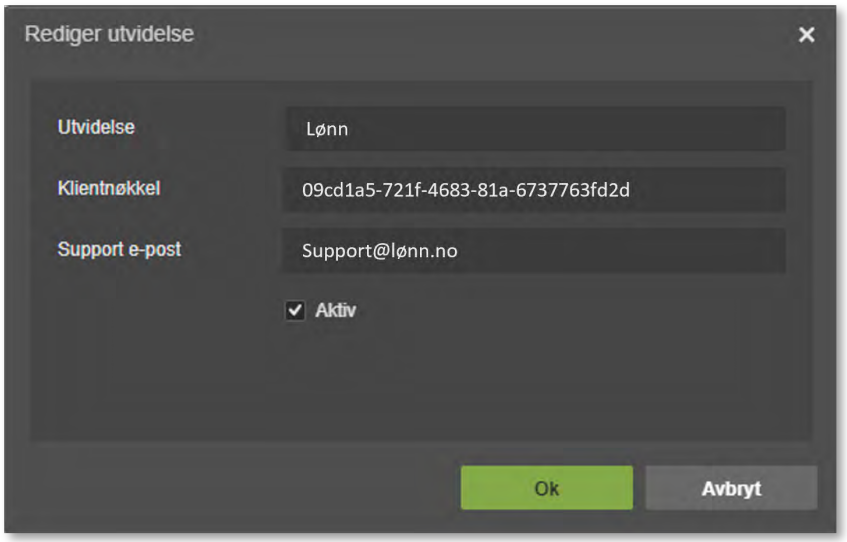

**Figur A2.1:** Første applikasjonsvisning for administrator.

Øverst til høyre i tabellen finnes en + knapp for å legge til nye klienter, denne åpner en dialogboks. LAGRE knappen aktiveres når klientnavn og klientnøkkel er fylt ut.




**Figur A2.2:** Dialogboks for ny klient.

Klientnøkkel for integrasjon mot Poweroffice GO hentes under utvidelser i GO. Legg til utvidelsen Lønn og kopier klientnøkkelen.

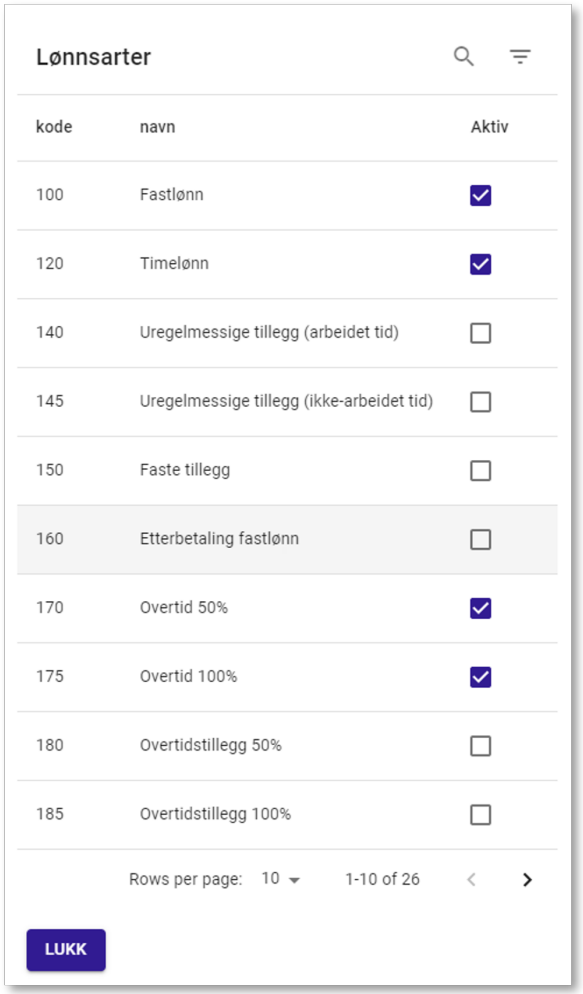



**Figur A2.3:** Legge til utvidelse i GO.

Når en ny klient opprettes lastes automatisk alle relevante lønnsarter inn fra GO. Det vil si alle lønnsarter med følgende behandlingsmåte:

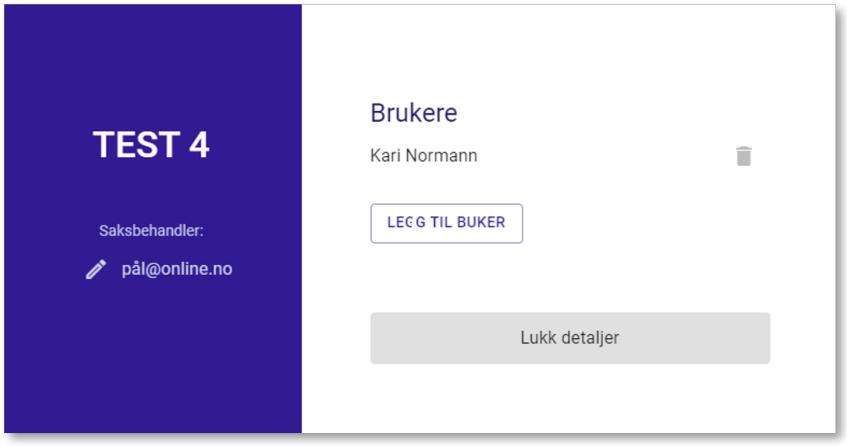
- FixedAmount
- FixedAmountWIthQuantity
- QuantityAndRate
- QuantityAndFixedRate
- FixedSalary
- HourlyWage
- PercentOgHourlyWage

En dialogboks med alle lønnsartene åpnes og administrator må deretter krysse av for hvilke lønnsarter som skal være tilgjengelig. Endringer lagres umiddelbart. Det er også mulig å endre lønnsarter senere.




**Figur A2.4:** Dialogboks for redigering av lønnsarter.

Når en ny klient er opprettet må brukertilganger legges til. Dette gjøres ved å trykke på REDIGER på aktuell klient, som vises ved å holde musepekeren på aktuell linje. Da åpnes en dialogboks hvor brukertilganger kan legges til og slettes og saksbehandlers e-post kan redigeres. Brukere må opprettes før de kan få tilgang til en klient, dette gjøres på siden for brukere som vises i figur A2.6.

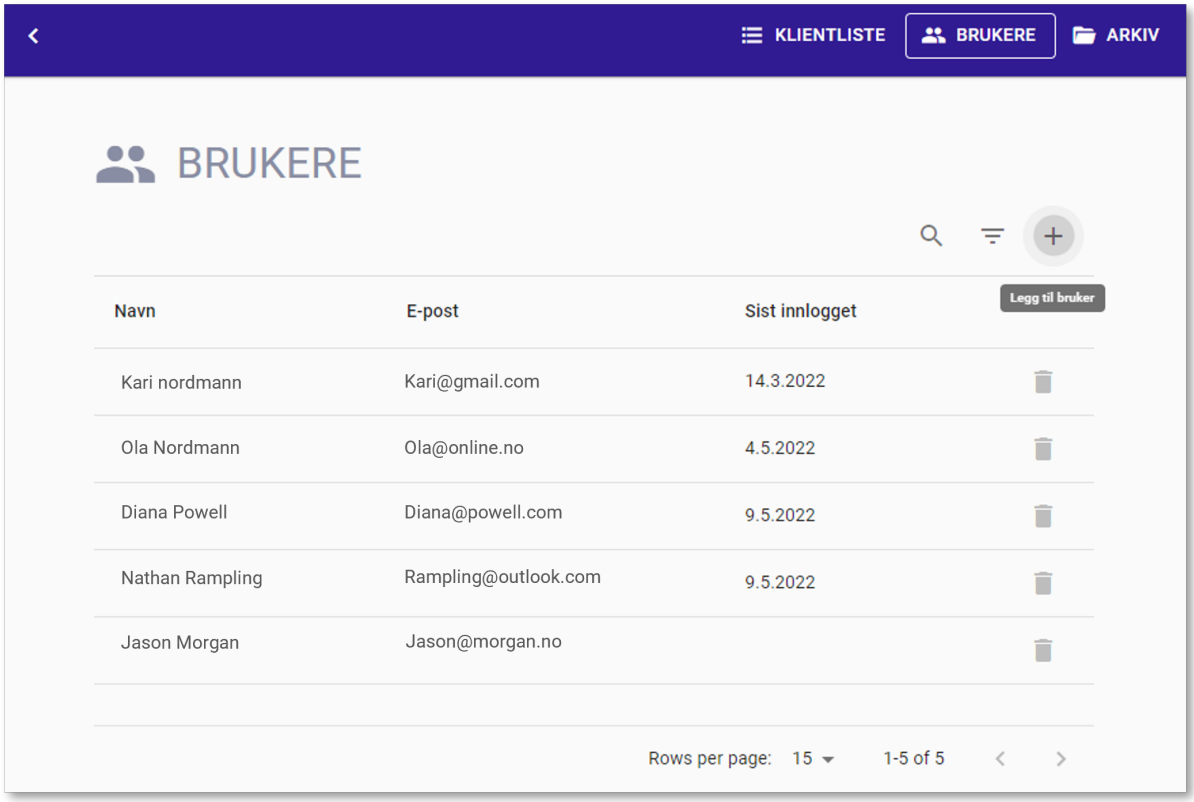



**Figur A2.5:** Dialogboks for å redigere brukertilganger og saksbehandler.

Det er også mulig å slette klienter, dette gjøres ved å trykke på på slutten av aktuell linje. En bekrefelsesdialog vil vises for å hindre sletting ved uhell. Dersom administrator ønsker å endre tilgjengelige lønnsarter for en klient kan det gjøres ved å trykke på LØNNSARTER. Lønnsarter oppdatereres fra GO og dialogboksen i figur A2.4 vises.

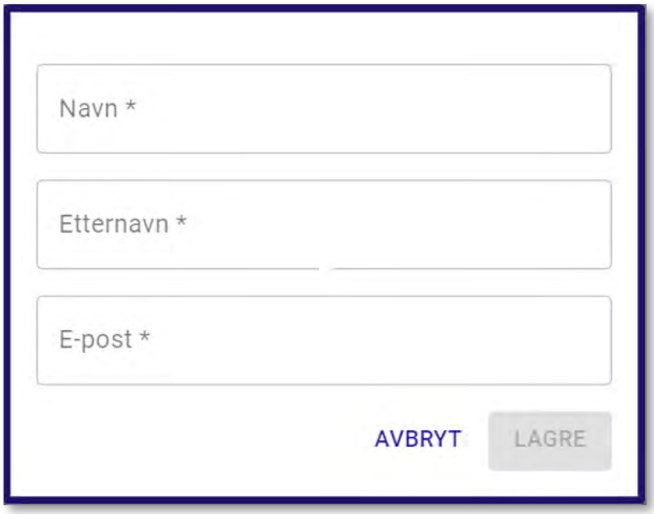
Øverst til høyre er det mulig å bytte mellom de ulike sidene Klientliste, Brukere og Arkiv.

På siden Brukere vises en liste over brukere, tilsvarende den for klienter. Det er mulig å slette brukere med .



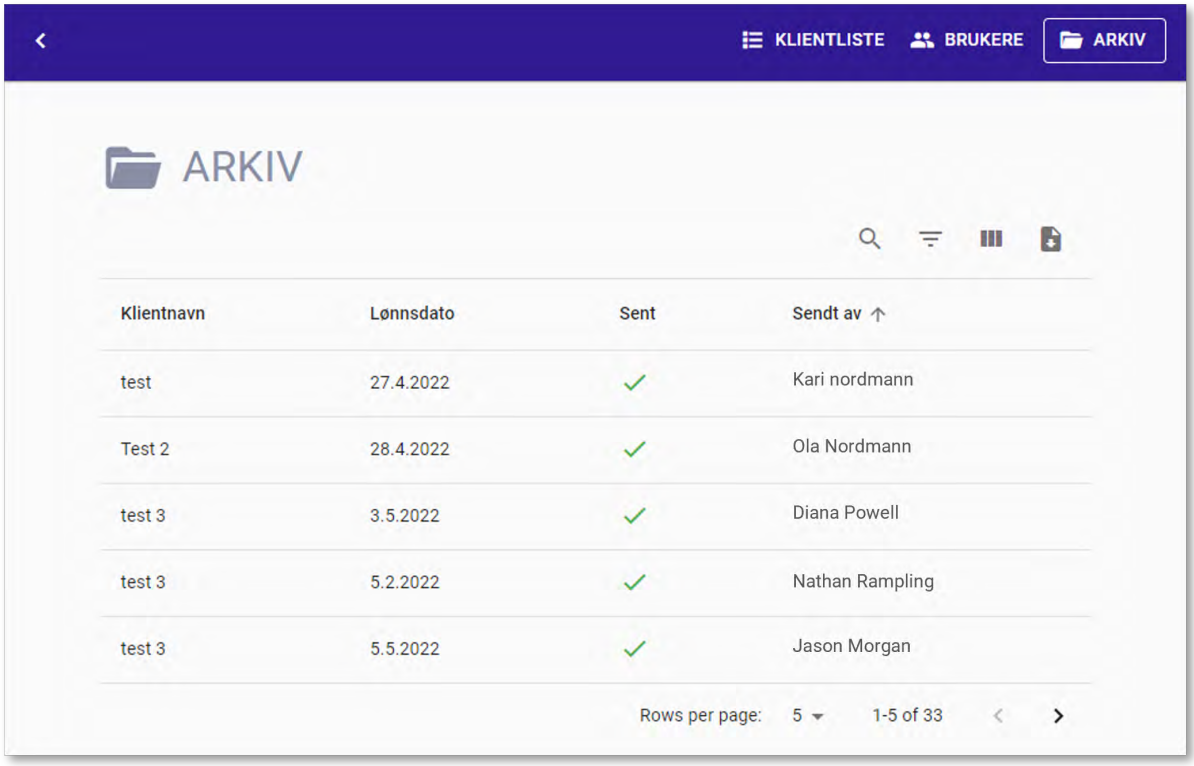

**Figur A2.6:** Liste over brukere.

For å legge til en ny bruker trykker man på + tegnet øverst til høyre i tabellen. Da vises en dialogboks hvor informasjonen legges inn. Alle feltene er obligatoriske og lagreknappen aktiveres når de er fylt ut.



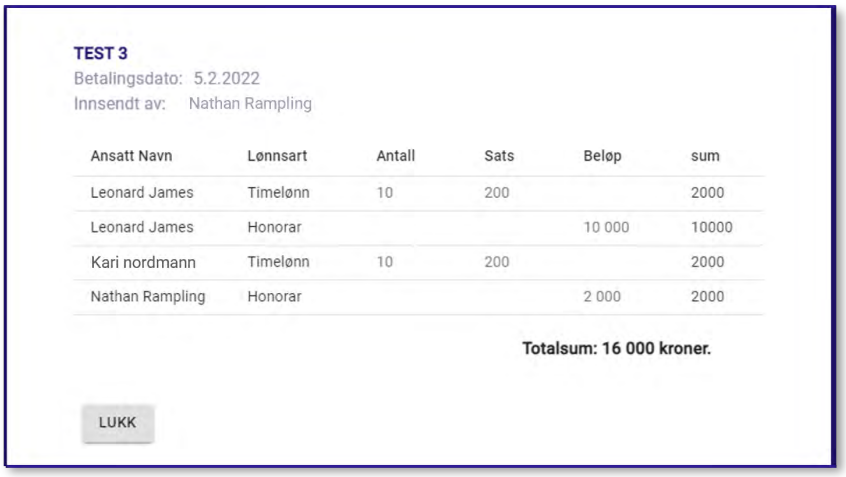

**Figur A2.7:** Dialogboks for å legge til bruker.

Siste siden administrator har tilgang til er Akriv. Her vises alle lønnsgrunnlag i løsningen med en ✓ dersom det er sendt. Informasjon om hvem som har sendt grunnlaget vises også.




**Figur A2.8:** Arkiv med innsendte lønnsgrunnlag.

Ved å trykke på en linje i lønnsarkivet vises en dialogboks med detaljene fra lønnsgrunnlaget.

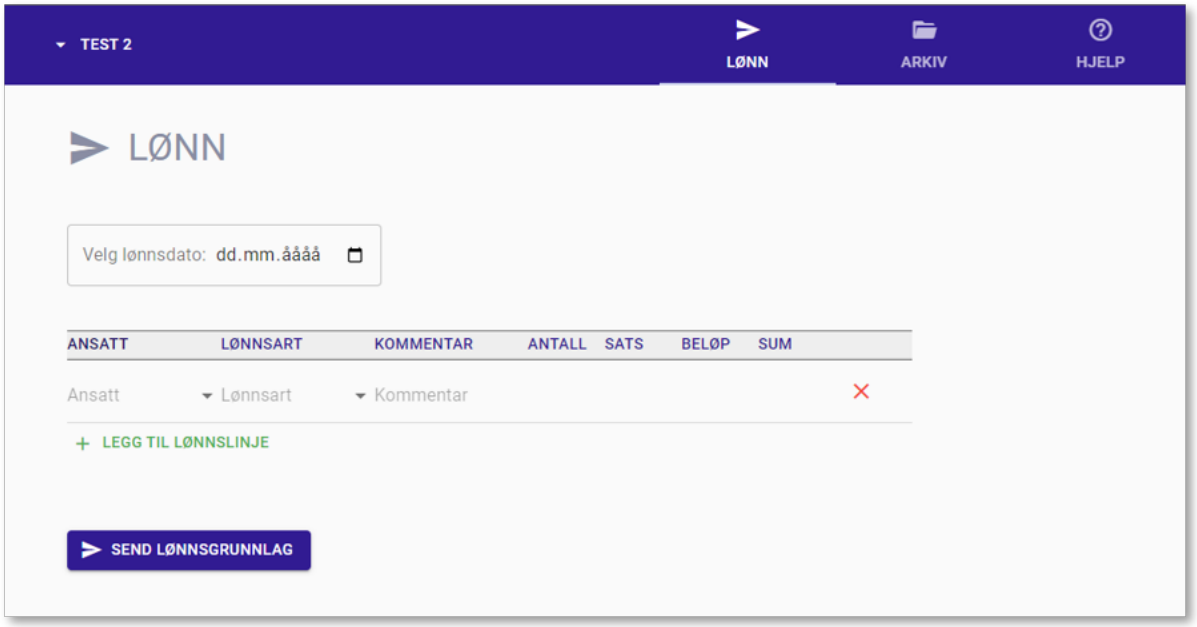



**Figur A2.9:** Detaljvisning av lønnsgrunnlag.

# A3 Applikasjon: Sluttbrukergrensesnitt

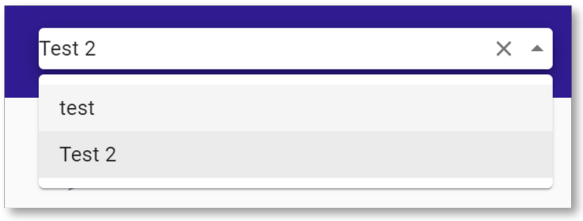
Ved innlogging er applikasjonsvisningen på siden Lønn. Figur A3.1 viser denne siden. Under overskriften er det en datovelger. Skriv inn betalingsdato eller velg dato ved trykke igjennom kalenderen som vises når en trykker på kalender-ikonet. Det er ikke mulig å velge forbigåtte datoer.

Trykk på + LEGG TIL LØNNSLINJE for å legge til lønnslinjer. Det er mulig å legge til flere lønnslinjer per ansatt. Velg ansatt i nedtrekksmenyen. Her vises alle ansatte tilhørende klienten. Lønnsart velges også fra en nedtrekksmeny. Det er valgfritt å legge inn en kommentar, men antall timer og sats eller beløp må fylles ut avhengig av lønnsart som er valgt. Trykk på ✕ for å slette en lønnslinje.



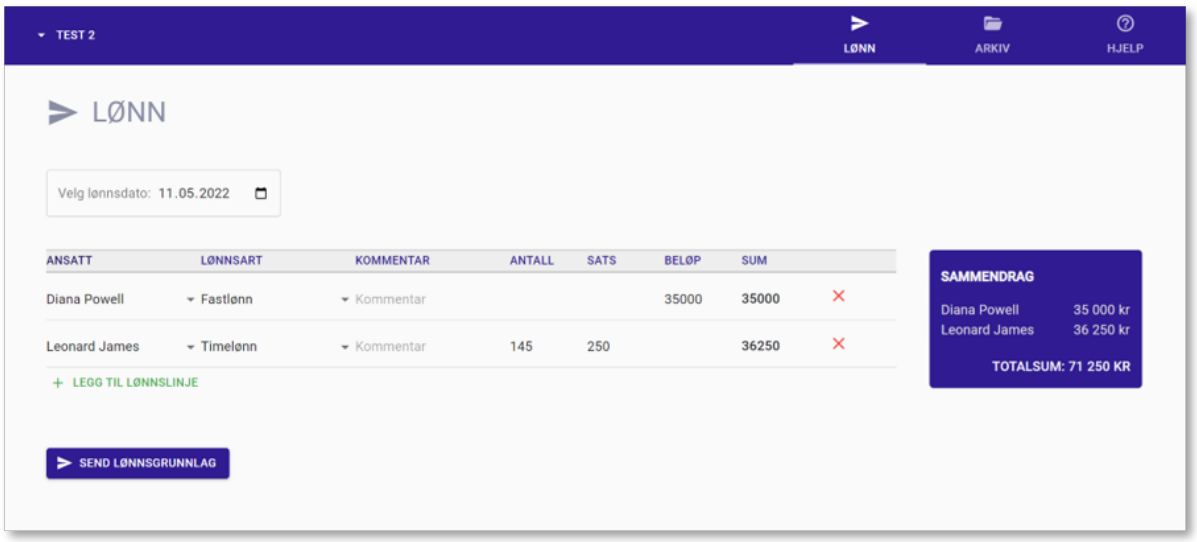

**Figur A3.1:** Første applikasjonsvisning for sluttbruker.

Oppe i venstre hjørne står det hvilken klient du er logget inn på. Ved å trykke på TEST 2 dukker det opp en nedtrekksmeny som vist i figur A3.2. Her kan du velge blant klientene du har tilgang til.



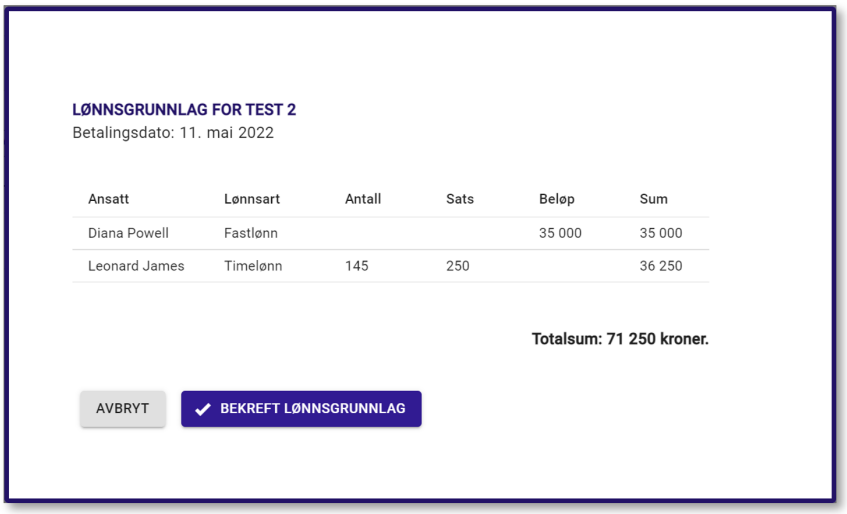

**Figur A3.2:** Sluttbruker kan velge klient.

Når du fyller på med lønnslinjer vil et sammendrag på ansattnivå vises til høyre på skjermen. For eksempel dersom en ansatt har flere linjer vil totalsummen for den ansatte vises til høyre for navnet.



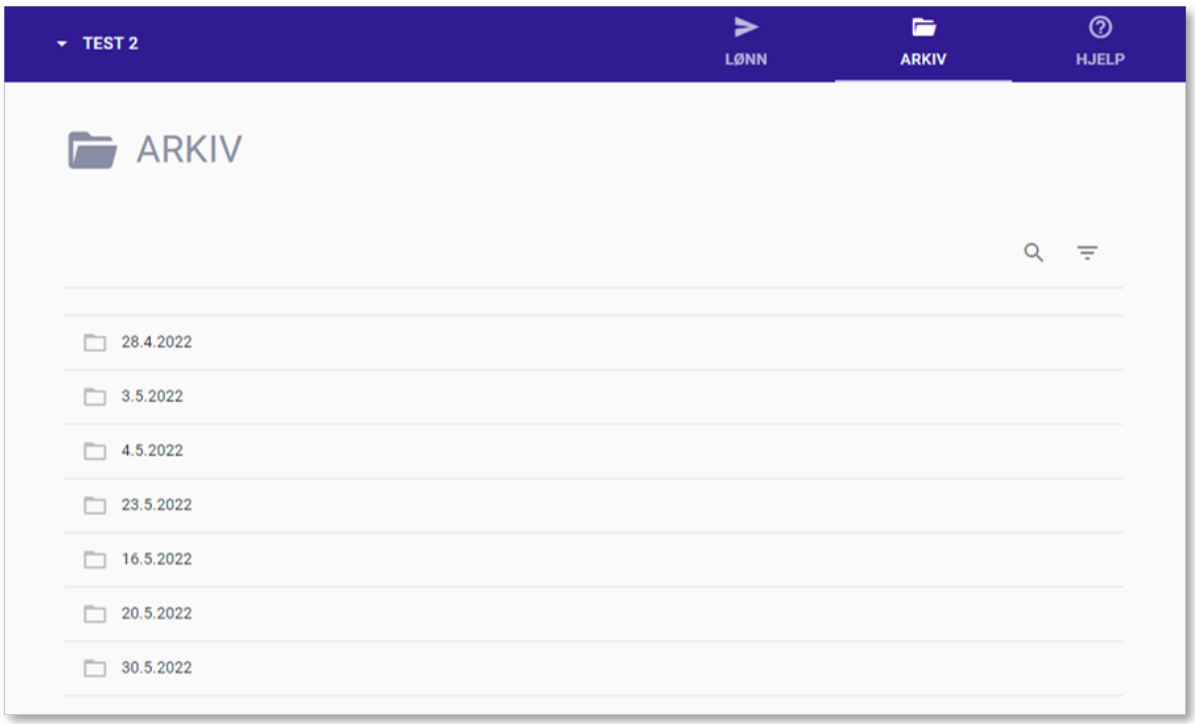

**Figur A3.3:** Utfylte lønnslinjer og sammendrag.

Det er ikke mulig å trykke på SEND LØNNSGRUNNLAG før det er valgt en dato. Før lønngrunnlaget sendes til GO fremkommer en dialogboks hvor du må se over og bekrefte lønnsgrunnlaget.



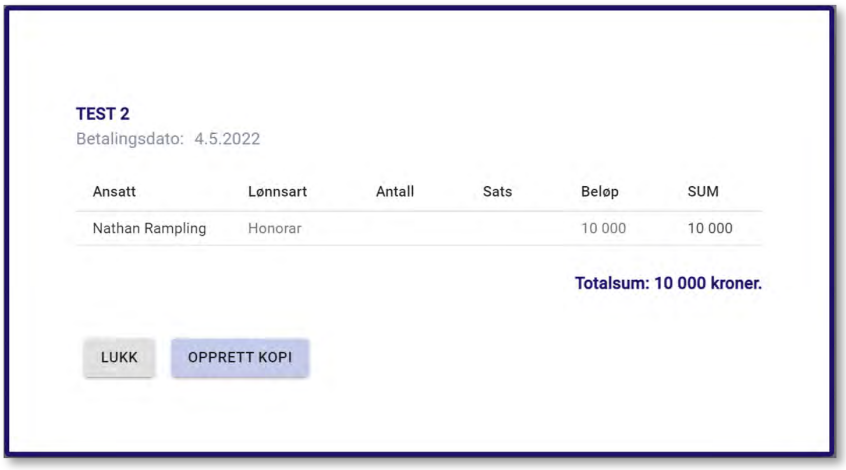

**Figur A3.4:** Bekreft lønnsgrunnlag.

Bytt mellom faner oppe til høyre. Arkivet vises som en liste med datoer for tidligere lønnsgrunnlag. Det er mulig å søke og filtrere i listen.




**Figur A3.5:** Arkiv listet etter dato.

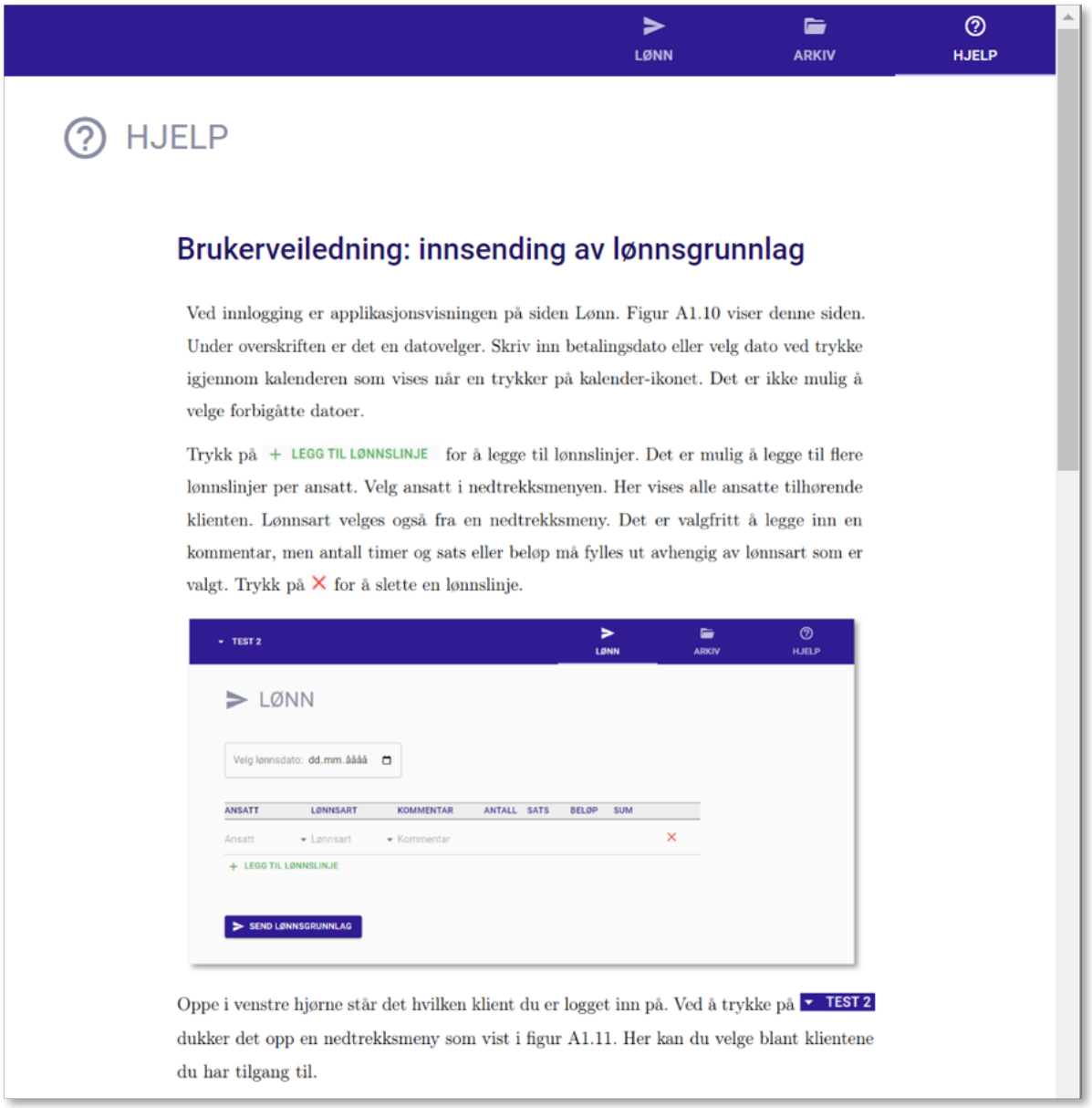
Trykk på radene for å få frem lønnsgrunnlagene som er arkivert på betalingsdato. Her er det mulig å kopiere det arkiverte lønnsgrunnlaget ved å trykke på OPPRETT KOPI.




**Figur A3.6:** Arkivert lønnsgrunnlag.

Når du trykker på OPPRETT KOPI vil applikasjonsvisningen automatisk laste inn siden for Lønn. Lønnslinjene fra Arkivet vil da være fylt ut. Her er det mulig å gjøre endringer på lønnslinjene om nødvendig.

Sluttbuker har også en fane for instrukser om hvordan applikasjonen fungerer slik som vist i figur A3.7.




**Figur A3.7:** Hjelpeside.