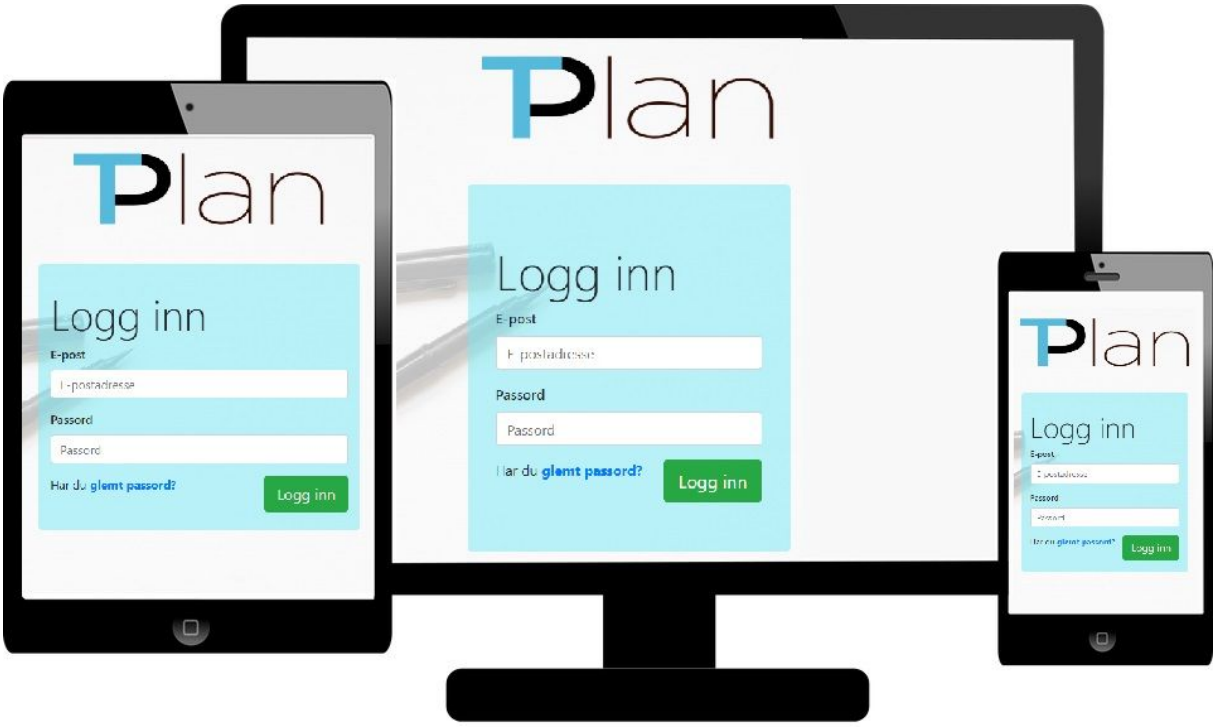
Universitetet i Sørøst-Norge



BOP3000R Bacheloroppgave

IT og Informasjonssystemer

24.05.2019

Roza Dawood Shakr | Elise Johnsen | Daniel Rajan | Majd Nakhleh

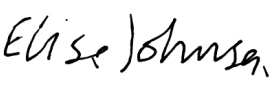
Forord

Denne bacheloroppgaven er skrevet av fire studenter i forbindelse med avslutningen av vårt bachelorstudium i IT og informasjonssystemer ved Universitetet i Sørøst-Norge, Campus Ringerike. Arbeidet med oppgaven har vært utfordrende, tidkrevende og spennende. Vi er stolt av sluttproduktet, og er fornøyd med prosessen som har ført oss hit. Oppgaven ga oss muligheten til å bruke kunnskap og ferdigheter som vi har tilegnet oss gjennom hele studiet samt muligheten til å lære noe nytt.

Gruppen vil gjerne takke vår veileder Salah Uddin Ahmed for gode råd og støtte. Vi føler at Ahmed har vært tilgjengelig for oss, gitt konstruktive og spesifikke tilbakemeldinger som har vært til stor hjelp.

Oppgaven hadde ikke vært mulig uten vår oppdragsgiver, Husam Salah. Vi ønsker å takke for oppdraget og for tålmodigheten gjennom utviklingsprosessen.






Roza Dawood Shakr Daniel Rajan Elise Johnsen Majd Nakhleh

Sammendrag

Denne rapporten beskriver arbeidet som ble gjort for å utvikle en webapplikasjon for en ekstern oppdragsgiver. Oppdraget er gitt av Husam Salah som er daglig leder for Pasta Cafe. Han forklarer at en vil trenge en løsning til administrering av lønn og turnus. Vi har i denne oppgaven utarbeidet og lagt frem de ulike stegene vi har tatt for oss for å kunne oppfylle kravene til oppdragsgiver. Videre også lagt frem hvorfor og hvordan vi har kommet frem til de forskjellige beslutningene.

Rapporten er bygd ut i flere deler, vi starter med planlegging, så videre til analyse, design og implementerings delen til slutt. I planleggingsdelen skal vi dokumentere drøftingen vår og hvorfor de ulike valgene ble tatt, vi skal dokumentere hva vi mener estimert tid skal være i de forskjellige delene i prosjektet og aktuell tid som ble brukt. Videre i analyse delen, beskriver vi blant annet bruksmønsterdiagram, målgruppe og alternative løsninger. I design-delen, har vi lagt ved blant annet wireframes av webapplikasjonen og til slutt går vi videre til siste fase som er implementeringsfasen, der vi beskriver hvordan vi har bygd systemet og hvordan prosessen ble gjennomført. Til slutt er det vår diskusjon og oppsummering av hele prosjektets prosess. Her tar vi opp ulike temaer som blant annet utviklingsmetoden vi benyttet og samarbeidet innad gruppen.

Vårt formål med denne oppgaven er å utvikle en dynamisk webapplikasjon som er brukervennlig, noe som krever god planlegging og design. Vi besluttet å bruke Scrum-metodikken som et fleksibel rammeverk for å utvikle dette prosjektet, denne metoden har mange fordeler som passer best med tanke på at utvikling og testing kan repeteres helt til ønsket sluttprodukt er oppnådd.

Innholdsfortegnelse

1. Introduksjon

Pasta Café er en kafé og restaurant som er lokalisert på Kuben kjøpesenter i Hønefoss sentrum. Bedriften driftes og eies av Husam Salah med rundt åtte ansatte. Han har drevet kaféen i over to år og har etablert seg faste rutiner og prosedyrer. Per dags dato bruker han papirformat for å framlegge vaktliste på arbeidsplassen, føre inn antall arbeidstimer for hver ansatt og før månedsskiftet må Salah sørge for å summere totale kostnadene som bedriften bruker på sine ansatte.

Ansatte må sørge for å holde seg oppdatert over sine kommende vakter, da vaktlisten blir skrevet ut og henges opp på arbeidsplassen. Når daglig leder fører opp timeliste, er det ansatte sitt ansvar til å korrigere feil eller mangel dersom de finner noe som ikke stemmer overens med føringer av timer eller lønn. Salah innser at dette er lite effektivt og upraktisk i lengden. Dettet krever mye tid og ressurser og kan før til kommunikasjonssvikt mellom han og de ansatte. Han ønsker et system som kan forenkle arbeidsoppgavene hans og informasjonsformidling til ansatte.

Formålet med dette prosjektet innebærer å utvikle en dynamisk webapplikasjon som brukerne har tilgang til uansett tid og sted. Applikasjonen skal kunne gi begrenset adgang avhengig av brukernes rolle. Ansatte får tilgang til oversikt over informasjon om deres vakter, kommende vakter, månedlig lønnsoversikt og flere funksjoner. Salah som er daglig leder vil få en rolle som administrator, det vil gi han mer rettigheter til å legge til, lese, endre og slette informasjon.

Vi ser at IT og informasjonssystemer er stadig i utvikling, og med dagens informasjonsteknologi kan vi realisere Salahs sitt formål. Applikasjonen skal gjøre arbeidshverdagen lettere, både for de ansatte og for Salah. Vi følger systemutviklings livssyklus og benytter oss av moderne utviklingsmetode, teknologi og verktøy til å lage en brukervennlig applikasjon som erstatter manuelle arbeidsoppgaver.

2. Bakgrunnslitteratur

I denne delen vil vi introdusere metoder, rammeverk, språk og verktøy som er benyttet i prosjektet. Vi beskriver hva som er brukt og hva det er brukt til.

2.1. Utviklingsmetode

Et utviklingsprosjekt består av strukturerte faser som gir god oversikt over prosjektet, det anbefales å utvikle informassjonssystemer etter systemutviklingens livssyklus, på engelsk System Development Life Cycle (SDLC). Systemutviklingens livssyklus består av fire faser som er planlegging, analyse, design og implementering. Hver fase inneholder teknikker som produserer et ønsket produkt, og figur 1 under viser hvilket *fokus*, utvalgte *teknikker* og *produkt* hver fase er tenkt å inneholde (Roth, Dennis, Wixom, 2013, s. 10).

Fase	Fokus	Teknikker	Produkt
Planlegging	Hvorfor utvikle dette systemet? Hvordan strukturere prosjektet?	● Tidsestimering ● Gantt skjema	Prosjektplan -Tidsplan - Arbeidsplan Risikoanalyse
Analyse	Hvem, hva, hvor og når for systemet	● Intervju ● Use case analyser	Systemkrav - Bruksmønster
Design	Hvordan skal systemet fungere?	● Valg av programvare og maskinvare ● Brukergrensesnitt - struktur og prototype ● Datamodellering ● Normalisering	Systemspesifikasjon - Systemarkitektur - Wireframes - Databasemodell
Implementering	Programmere Levering og støtte av ferdig system	● Webutvikling ● Administrasjon av database	Dokumentasjon

Figur 1, Systemutviklingens fire faser

Planlegging

Under planleggingsfasen skal man stille seg tre spørsmål; Kan vi bygge det? Vil det gi bedriften verdi? Hvis vi bygger det, vil det bli brukt? Utredningen kartlegger om prosjektet bør igangsettes. Når et prosjekt er godkjent utarbeides det en arbeidsplan og en prosjektplan som beskriver hvordan prosjektgruppen vil jobbe med prosjektet.

Analyse

I analysefasen skal man svare på hvem som vil bruke systemet, hva systemet skal gjøre, og hvor og når det vil bli brukt. For å utvikle konseptet for det nye systemet går man gjennom tre steg. Først analyseres det eksisterende systemet og identifisere hvilke forbedringer man ser for seg i det nye systemet. Deretter samles inn systemkrav, ofte gjennom intervju av de som skal bruke det nye systemet. På bakgrunn av dette arbeidet, lages modeller som viser hvilke data og prosesser systemet er tenkt å støtte.

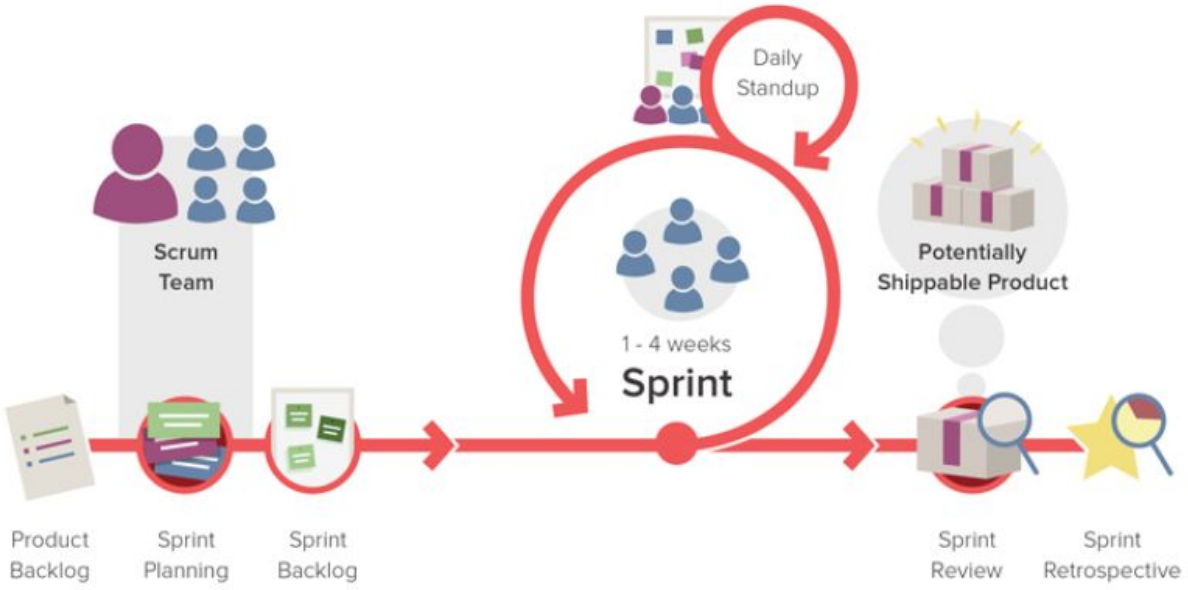
Design

I denne fasen bestemmes hvordan systemet skal fungere med tanke på maskinvare, programvare, infrastruktur og nettverk som resulterer i systemarkitektur. Man lager også et design for brukergrensesnitt som viser hvordan brukere vil navigere i systemet. Man spesifiserer hvordan data vil bli lagret i database, til sammen utgjør samlingen av produkter en systemspesifikasjonen.

Implementering

Her utvikles og feilsøkes systemet og man installerer det hos bruker og støtter brukeren i oppstarten. For omfattende systemer kan det være aktuelt å utvikle et treningsprogram, eller kurs, som brukerne skal gjennom (Roth, Dennis, Wixom, 2013, s. 10).

Scrum



Figur 2, Scrum

Scrum er et agile rammeverk for samarbeid i grupper og produktutvikling. Med Scrum er teamet selvorganiserende og velger selv hvordan de vil jobbe for å skape mest mulig verdi og effektivitet. I Scrum jobber man i *Sprinter*. Sprintene gjør det mulig å effektivisere arbeidet, slik at teamet blir ferdig med å levere et ferdig produkt til kunden (glasspaper, u.å.). En Sprint består av *Sprintplanlegging*, *Daglig Standup*, *Sprint-review* og *Sprint Retrospective*.

Sprintsplanleggingen er først fase i et Sprint, under planleggingen velger teamet hvilke funksjoner fra systemkravene som skal gjøres i denne Sprinten. *Daglige Standup* er korte møter, der hver gruppemedlem forteller hva de har gjort, hva de skal gjøre, og eventuelt om de møtte på problemer. Siste dag i sprinten, skal teamet holde *Sprint Review,* hvor teamet presenterer resultatet av Sprinten til kunden og gir hverandre konstruktive tilbakemeldinger. I *Sprint Retrospective* går de gjennom prosjektets prosess, deretter starter sirkelen på nytt (Helle, 2013).

Teknikker og produkter

Gantt-skjema er et horisontalt søylediagram som viser oppgavene i arbeidsplanen. Det gir et visuelt overblikk over oppgavenes forhold til tidspunkt vi har satt til hver fase (Roth, Dennis, Wixom, 2013, s. 94).

Risikoanalyse er en oversikt over problemer som kan oppstå i prosjektet, og hvilke konsekvenser de kan forårsake. Risikoanalyse kan bidra til redusering av risiko for problemer med prosjektet (Aven, 2016).

Funksjonelle krav til et system kan beskrives med bruksmønster, på engelsk *use case*. Et bruksmønster beskriver en sekvens interaksjoner mellom aktør, brukeren, og systemet for å oppnå et mål (Skagestein, 2002).

Systemarkitektur viser strukturen til systemer og viser hvordan komponentene er koblet til hverandre. Systemarkitekturen er tenkt å gi en oversikt over applikasjonen og hvordan data sendes mellom de ulike elementene (Spacy, 2018).

Wireframes er oppsettet til en webside som viser hvordan de ulike grensesnittelementene er plassert i forhold til hverandre. Wireframes brukes tidlig i utviklingsprosessen for å raskt og enkelt etablere en grunnleggende struktur for applikasjonen. (Experience, u.å)

2.2. Prinsipper og uttrykk

Universell utforming

Universell utforming av IKT systemer handler om å utforme løsninger slik at flest mulig, uavhengig av funksjonsevnen til den som skal kunne benytte seg av løsningen (Difi, 2019). IKT-løsninger for arbeidslivet er unntatt av forskriften som normalt gjelder nettløsninger og applikasjoner (Difi, 2018). Derfor er ikke vi nødt til å oppfylle de kravene som forskriften setter, men vi ønsker likevel å utvikle en løsning som fungerer for mange, og vi vil benytte oss av prinsippene bak universell utforming. Designet og utformingen av applikasjonen skal være lett å forstå uavhengig av de ansattes kunnskap eller erfaring. Webapplikasjonen skal

kunne kommunisere nødvendige informasjon til de ansatte på en effektiv måte (Sandnes, 2011, s.28). Utformingen av applikasjonen må tilpasse ulike skjermstørrelser slik at brukere får en god brukeropplevelse også på mobil. Samtidig skal systemet være ryddig, lett å navigere og inneholde lite scrolling (Adolfsen, 2015).

Navigasjon

Navigering er en del av brukergrensesnittet og det skal være mulig å ha ulike metoder for å navigere seg rundt i systemet. Ofte bruker nettsider to metoder som meny eller søkefunksjon for navigering. Det er lite hensiktsmessig å ha et system som har et godt innhold, men tungt å navigere seg rundt (Difi, 2015). Menyen bør skille seg ut fra innholdet på siden og skal være lett tilgjengelig, slik det ikke skaper frustrasjoner. Vi skal bruke «sticky meny», det vil si at menyen sitter fast på siden, slik at når brukeren scroller nedover er menyen alltid lett tilgjengelig (Gangåssæter, 2016).

Layout

Layout handler om å plassere tekst, skrift, bilde og annet innhold på en strukturert måte. Videre er det viktig med å skape blikkfang fra brukere, det kan man oppnå ved hjelp av farger, størrelse på skrift, bilder og lignende. I tillegg er det viktig å ha en god balanse mellom farger, skrift og bilder for å redusere støy (Gundersen, 2017). Å bruke riktige farger er viktig, fordi ulike farger kan bety ulike ting, for eksempel rød kan enten beskrive fare eller kjærlighet. Dette har selvsagt med kultur og miljø å gjøre, ikke nødvendigvis med den målgruppen man skal holde seg til. Det er viktig at lenker er merket med understrek og har en annen farge. Fargene på en nettside er en av de første brukere legger merke til, å ha en god kombinasjon mellom farger og tekst er hensiktsmessig for å ikke ta fokuset deres fra den nødvendige informasjonen (Difi, 2015).

2.3. Database

Database er "en samling logisk relaterte data som kan brukes for et bestemt formål" (Kristoffersen, 2016, s. 3). Flertallet av databasesystemer i dag benytter seg av

relasjonsdatabaser. I relasjonsdatabaser er data lagret i tabeller med relasjon til hverandre via *nøkler* som er unike identifikatorer som for eksempel "identifikasjonsnummer" eller lignende (Kristoffersen, 2016, s. 4). I denne applikasjonen er det viktig med en database, fordi innholdet i applikasjonen er data som må lagres og være lett tilgjengelig.

Normalisering av en database vil si å fjerne unødvendige redundans, altså overflødig informasjon. Dette er viktig for å spare lagringskapasitet. Et tilfelle kan være at flere ansatte har samme postnummer, som igjen betyr at de har samme poststed. Istedenfor å lagre alle ansatte med postnummer og poststed lager man en egen tabell for denne informasjonen og legger kun ved postnummer i ansatt tabellen. Ved hjelp av postnummer kan man "slå opp" poststed i den andre tabellen og man spare unødvendig lagring av data.

SQL-planting

SQL-planting er et form for angrep hvor hackeren bruker åpne skjemaer på applikasjonen for å manipulere meldingen som blir sendt til databasen. Man kan for eksempel sende en melding som tilsier at systemet skal se bort ifra at påloggingsinformasjonen mangler riktig passord, eller man kan i verste fall endre meldingen som sendes slik at hele databasen slettes. (Horgen, 2014, s. 300). PHP har støtte for en teknikk som kalles "prepared statements" som sjekker dataen som kommer inn, og håndterer eventuelle uregelmessigheter (Horgen, 2014, s. 313).

2.4. Utviklingsspråk

Åpen kildekode er et begrep som omhandler programvare som distribueres med kildekoden er tilgjengelig for brukerne. Motsetningen er kildekode som distribueres i binær form og er hemmelig, og ofte patentbeskyttet. (Gramstad, 2018).

HTML, eller HyperText Markup Language er et *markeringsspråk* for å organisere, og strukturere, informasjon på en nettside. Man kan definere seksjoner, tabeller og skjemaer og om tekst er en overskrift, en liste og til å plassere media som bilder og video på siden (Felke-Morris, 2012).

CSS står for *Cascading Style Sheets* og er et språk som beskriver hvordan HTML elementene blir designet på en nettside (W3schools, 2019). Vi har brukt vår css-stil sammen med Bootstrap ferdige css-stil.

PHP står for HyperText Preprocessor og er et programmeringsspråk som benyttes for å lage interaktive, dynamiske nettsteder (Horgen, 2014). Noen av fordelene med PHP er at språket har innebygde funksjoner for kommunikasjon med databasen, og "hashing" og "salting" av passord. Hashing av passord er en metode hvor en algoritme gjør om passordet til en ny tilfeldig tekststreng (Horgen, 2014, s. 322). Dermed er det ikke mulig å lese passordet i klartekst. "Salting" er en metode å "krydre" passordet med en ekstra tekststreng, saltet, for å lage et lengre og mer komplisert passord, som er vanskeligere å knekke (Horgen, 2014, s. 327).

SQL, Structured Query Language, er et språk som er laget for å jobbe med databaser. Man bruker SQL for å lage spørringer for å hente ut, eller sette inn, data i databasen. (Kristoffersen, 2016, s. 4)

2.5. Rammeverk og verktøy

Denne rapporten er skrevet i Google Docs, et tekstbehandlingsprogram som gjør det mulig for alle gruppemedlemmene å skrive i dokumentet samtidig.

For å kommunisere har vi hovedsakelig brukt en gruppechat i Facebook Messenger. Vi har prøvd ulike kommunikasjons- og prosjektstyringsverktøy tidligere, men har gjennom studiet gjort erfaringer med at "det enkle er ofte det beste" og man faller tilbake på Google Docs og Facebook Messenger som gode verktøy med lav terskel for å bruke.

MySQL er et databasesystem som støtter SQL og er en åpen kildekode (Horgen, 2014, s. 241). Vi brukte MySQL sammen med MySQL Workbench, som et verktøy for å administrere databasen.

GitLab er en gratis applikasjon som gjør det enkelt for team å samarbeide med prosjektet. Ved hjelp av GitLab kunne vi administrere, opprettet, overvåke og dele kodene våre med hverandre (GitLab, 2019).

Visual Studio Code (VSC) er et kraftig programmeringsspråk verktøy. Dette programmet er gratis å bruke og støtter macOS, Windows og Linux plattformer. Vi benyttet dette programmet for å kode websiden vår. Samtidig kan man også koble det mot GitLab, slik at kodene er alltid synkronisert (Microsoft, 2019).

Webhotell er lagringsplass for websider på en server som alltid er tilgjengelig via Internett. Webhotell knyttes til et domene, i vårt tilfelle tplan.no slik at andre kan finne webapplikasjonen vår (Proffhosting, u.å.).

Visual Paradigm er et modelleringsverktøy som man kan bruke til å planlegge og dokumentere et system ved å lage, blant annet, modeller, og i vårt tilfelle wireframes.

Bootstrap er et rammeverk for utforming av nettsteder. Bootstrap ble opprinnelig utviklet av Twitter, men som ligger tilgjengelig for alle som åpen kildekode (Otto & Thornton, 2019). En av mange fordelene til Bootstrap er at innholdet blir justert etter hvilke skjermstørrelser som blir brukt.

3. Prosjektgjennomførelse

I dette kapittelet dokumenterer vi hvordan vi gjennomførte de ulike fasene av systemutviklingens livvsyklus; planlegging, analyse, design og implementering. Kapittelet starter med en oppsummering av arbeidet i disse fasene.

Kontraktsinngåelse bestod av å danne gruppen og møte med oppdragsgiver. Her fikk vi en beskrivelse av hva oppdragsgiver ønsket av oss, og kravoversikt med tanke på hva oppdragsgiver legger mest vekt på. Vi måtte fremlegge oppdraget for studiekoordinator for å få oppgaven godkjent. I denne innledningsfasen brukte vi også tid på å opprette en kontrakt som er inngått mellom gruppen, oppdragsgiver og universitetet. I tillegg til dette, skrev vi en kontrakt innad i gruppen (se vedlegg 8.8) som gjelder for arbeidsregler og andre regler for å effektivisere gruppearbeidet.

Når kontraktsinngåelsen var fullført gikk vi over til å planlegge prosjektet. Alle gruppemedlemmer deltok for å kunne beskrive hva vi trengte for å utføre prosjektet og hvordan vi kan oppnå best mulig sluttresultat. Her satte vi også opp tidsfrister for når de ulike fasene skal være gjennomført.

For å kunne lage en realistisk arbeidsplan startet vi med innleveringsdato og jobbet oss bakover. I denne planen ser vi for oss at vi møtes to ganger i uken der vi jobber åtte timer per dag, i tillegg til to timer arbeid på egenhånd. Ofte så vil oppdraget kreve mer tid, men dette er minimum beregning for å kunne gjennomføre prosjektet.

I analysefasen måtte vi samle informasjon om hvem produktet er rettet mot og hvordan vi antar at de vil bruke det. I denne delen gjorde vi rede på hvilke krav og ønsker oppdragsgiver har og vi måtte vurdere om kravene er oppnåelige. Her møtte vi oppdragsgiver for å sikre oss at vi har fått god nok informasjon om de forskjellige kravene. Forståelsen av systemkravene er viktig for å kunne utvikle et godt produkt.

Designfasen bestod av å utføre arkitekturdesign, databasemodellering og wireframes. Arbeidet som ble gjort i denne fasen hjalp oss med å effektivisere utviklingsfasen. Når designfasen var gjennomført kunne vi gå videre til utvikling.

I implementeringsfasen jobbet vi mye i fellesskap. Dette gjorde at vi kunne hjelpe hverandre dersom en av oss møtte på problemer og trengte hjelp eller innspill. Denne fasen har fått en stor del av tiden vi hadde til rådighet i prosjektet. Dette var fordi vi ønsker god tid til å utvikle et produkt som vi var fornøyd med og god tid til å teste det og eventuelt rette opp feil eller mangler.

3.1. Planleggingsfasen

I planleggingsfasen skal man velge utviklingsmetode og lage en prosjektplan som består av blant annet tidsestimering, arbeidsplan og risikoanalyse.

3.1.1 Valg av utviklingsmetode

Noe av det første man gjør i arbeidet med en prosjektplan er å velge utviklingsmetode. Roth, Dennis og Wixom introduserer syv forskjellige metoder, med ulike styrker og svakheter (Roth, Dennis og Wixom, 2013, s. 51). Det var kun to av disse som var aktuelle for oss; fossefall og såkalte "agile" utviklingsmetoder. Disse to er på flere måter stikk motsatte av hverandre. Fossefallsmetoden karakteriseres ved at fasene i et prosjekt kommer sekvensielt etter hverandre, og man kan i utgangspunktet ikke gå bakover i fasene uten å starte på første fase. I følge fossefallsmetoden kan ikke et produkt leveres til kunden før hele systemet er ferdig utviklet. Agile metoder fokuserer på å levere deler av systemet til sluttbruker så tidlig som mulig for å få tilbakemelding, og deretter jobbe videre med flere deler av systemet etterpå. Det er seks viktige kriterier man kan bruke for å velge utviklingsmetode.

1. Usikkerhet rundt systemkrav
2. Kjennskap til teknologien
3. Kompleksiteten til systemet
4. Påliteligheten til systemet
5. Varigheten på prosjektet
6. Oversikt over tidsplan

Fossefallsmetoden blir omtalt som "God" på bare to av disse kriteriene; Systemer som er komplekse og som må være pålitelige. Agile kategoriseres som "Utmerket" på to punkter;

utvikling av systemer med uklare systemkrav og prosjekter med kort varighet. Samtidig er den "god" på utvikling av systemer som må være pålitelige og metoden gir god oversikt over tidsplanen (Roth, Dennis og Wixom, 2013, s. 59).

Vi skal utvikle et system, en applikasjon, som ikke er spesielt komplisert, vi har god kjennskap til teknologien som vi ønsker å benytte oss av, og vi vet at varigheten til prosjektet er kort. Det er i utgangspunktet ikke stor usikkerhet rundt systemkravet, men oppdragsgiver skal ha muligheten til å foreslå endringer etter hver sprint. På bakgrunn av dette ønsker vi å benytte oss av en *agile* utviklingsmetode, nemlig Scrum. Vi kommer tilbake til Scrum i implementeringsfasen hvor metoden blir benyttet til selve systemutviklingen.

3.1.2 Tidsplan

En metode for å estimere tidsbruk i de forskjellige fasene er å se på hvor lang planleggingsfasen er. Tanken er at et enkelt prosjekt har kort planleggingsfase og de andre fasene vil reflektere dette (Roth, Dennis og Wixom, 2013, s.62). Vi har satt av fire uker til planleggingsfase, som i teorien skal utgjøre 15% av prosjektets lengde. Med disse tallene burde analysefasen være på fem uker og designfasen ni uker og implementeringen åtte uker. Siden vi har valgt Scrum som utviklingsmetode vurderte vi at denne tidsfordelingen ikke stemte overens med vårt prosjekt. Ved bruk av Scrum legger man en del designarbeid i implementeringsfasen. Dessuten skulle vi ikke intervjue ansatte og kartlegge bruken av nåværende system, som er en vanlig del av analysefasen. Derfor har vi kortet ned disse to fasene og valgt å sette av mer tid til implementering, altså utvikling og testing av applikasjonen, da vi tenker at det er i denne fasen vi trenger mest tid. Figur 3 viser tidsplanen med de ulike fasene og planlagt start og varighet.

Aktivitet	Plan start	Plan uker varighet	Antall timer
Kontraktsinngåelse	25. november	1/2	36
Planlegging	1. Januar	4	288
Analyse	29. Januar	3	216
Design	19. Februar	3	144
Implementering	5. Mars	10	720
Ferdigstilling av rapport	14. Mai	2	144
Innlevering rapport	24. Mai		

Figur 3, Tidsplan

3.1.3 Arbeidsplan

En arbeidsplan er en dynamisk oversikt over de oppgavene som skal gjøres iløpet av prosjektet. For å utarbeide en arbeidsplan må man identifisere arbeidsoppgaver, og anslå hvor lang tid det vil ta å gjennomføre de. Ifølge Roth, Dennis og Wixom skal man også utarbeide en egen bemanningsplan, men vi har valgt å slå sammen denne med arbeidsplanen (Roth, Dennis og Wixom, 2013, s. 63). Vi følte ikke at det var nødvendig med to planer med tanke på omfanget av prosjektet vårt.Vi brukte teorien rundt SDLC, og de produktene som det er meningen man skal produsere i løpet av et prosjekt som mal for å identifisere arbeidsoppgaver. For å estimere tidsbruk, og rekkefølge, til hver oppgave brukte vi egne erfaringer fra tidligere prosjekt. Figur 4 viser arbeidsplan med oppgaver og hvem som er ansvarlig. Den viser også når oppgavene starter og ender.

ID	Navn	Varighet	Start	Slutt	Ansvarlig
	Analyse	**1.Feb - 25. Feb**			
1	Problemstilling / systemkrav		1. Feb	8. Feb	Majd
2	Prioritere krav		8. Feb	15. Feb	Majd
3	Utarbeide bruksmønster		1. Feb	25. Feb	Roza
4	Utarbeide datamodeller		1. Feb	25. Feb	Elise
	Design	**25. Feb - 5.Mars**			
5	Systemarkitektur		25. Feb	5. Mars	Daniel
6	Brukergrensesnitt / wireframes		25. Feb	5. Mars	Roza
7	Databasemodell		25. Feb	5. Mars	Elise
	Implementering	**5.Mars - 13. Mai**			
8	Lage database		5. Mars	25. Mars	Elise
9	Programmere		5. Mars	23. Apr	Daniel
10	Systemtesting		23. Apr	7. Mai	Roza
11	Skrive brukermanual		1. Mai	7. Mai	Elise
12	Installere hos oppdragsgiver		6. Mai	7. Mai	Majd
13	Opplæring oppdragsgiver		9. Mai	10. Mai	Majd
	Ferdigstilling rapport	**13. Mai - 24.Mai**			

Figur 4, Arbeidsplan

3.1.4. Risikoanalyse

Vi har satt opp risikopunkter som står sentralt i prosjektet, og hvilke tiltak som kan settes i verk for å redusere problemene som kan oppstå underveis. Punktene er hentet fra tidligere prosjekt og erfaringer vi har tatt med oss. Arbeidet med risikoanalysen ga oss en god diskusjon og resulterte i en plan for hva vi skal gjøre om vi opplever problemene som punktene beskriver. Utført risikoanalyse kan finnes som vedlegg 8.3.

3.2. Analysefasen

I analysefasen skal man gjennom tre steg for å forstå sluttbrukernes behov i det nye systemet. De tre stegene er:

- Forstå nå-situasjonen.
- Identifisere forbedringer
- Definere krav for det nye systemet (Roth, Dennis, Wixom, 2013, s.102).

Dette arbeidet ble gjort gjennom intervju med oppdragsgiver, en analyse av målgruppen og ved å se på alternativene som allerede finnes. Gjennom intervjuet med oppdragsgiver ønsker vi å kartlegge bedriftens behov og ønsker, samt å forstå hvordan oppgavene blir gjort manuelt i dag. Forståelse av målgruppen, altså sluttbrukerne våre, vil være avgjørende for hvordan vi designer applikasjonen. Ved å se på alternative applikasjoner vil vi kunne identifisere forbedringer vi kan implementere i vårt eget system. Arbeidet i analysefasen resulterer i en liste med systemkrav (Roth, Dennis, Wixom, 2013, s.103). Fullstendig liste over systemkravene finnes i vedlegg 8.4 og en oversikt over prioritert systemkrav finnes i Figur 5.

3.2.1. Målgruppe

I dette prosjektet er det viktig å definere målgruppen som skal benytte seg av applikasjonen fordi det blir enklere å utvikle en løsning som kan dekke deres behov. Målgruppen består av mennesker fra 16-30 som har travle og fleksible arbeidsdager. Dette er individer som trenger mer flyt og forutsigbarhet i arbeidsdagen og derfor kan systemet benyttes akkurat som brukeren ønsker. Vi antar at denne målgruppen som hovedsakelig består av unge mennesker vil bruke webapplikasjonen hovedsakelig på mobil plattform.

Gjennom intervju med produkteier har vi kartlagt nå-situasjonen og fått et inntrykk av ønsker til forbedringer i det nye systemet. På et overordnet plan er hensikten med det nye systemet å effektivisere administrasjon oppgavene til oppdragsgiver, og forenkle kommunikasjonen med ansatte. Gjennom webapplikasjonen skal brukere raskt kunne opprette og endre vaktlisten. Den skal også være brukervennlig og skal kunne brukes uten noe form for

bakgrunnskunnskap. Med denne webapplikasjonen gjør vi det enkelt for ansatte blant annet å få en god oversikt over deres vaktliste, foreslå endringer, inn og ut stemplingsur.

3.2.2. Alternative løsninger

For å forstå bedre nå-situasjonen har vi valgt å se på noen alternative løsninger til oppdragsgivers problem, altså andre applikasjoner som vi tenker at han kunne valgt å investere i istedenfor å gi oss oppdraget.

I dag finnes det flere alternativer av andre applikasjoner med samme formål som webapplikasjonen vår. En av de som vi har erfaring med fra tidligere jobber er Planday.

Formålet med denne Planday er å kunne gi bedrifter muligheter ved å gjøre forhold mellom medarbeiderne og ledere mer effektivt og produktivt. De har seks hoveddeler i deres produkt:

- Vaktplanlegging
- Kommunikasjon
- Lønnsberegning
- Personalhåndtering
- Timeregistrering
- Rapportering

Prisene er varierende fra bedrift til bedrift, men kostnadene er 40 kr per bruker per mnd + 500 kr per mnd. Hvis oppdragsgiver har 8 ansatte inkludert seg selv, må bedriften legge ut 820 kr mnd og 6,320 kr årlig. (Planday, u.å.)

Forskjellen mellom TPlan og Planday er at oppdragsgiver har vært med på å utforme TPlan slik at den tilpasset etter hans behov. I tillegg er TPlan kostnadsfri, hvis man ser bort fra kostnaden ved å ha den hos Domeneshop. Fordelen med alternative løsninger som Planday er at de er godt etablert og prøvd. Det er vår erfaring at det er slik med all kjøp av programvare. Etablert programvare som man kan kjøpe som hyllevare kan være vanskelig å tilpasse bedriften og man må ofte endrer hvordan man gjør ting for å tilpasse seg programvaren.

3.2.3. Systemkrav

Vi lagde en liste over systemkrav som er delt opp i funksjonsområder, som for eksempel *administrere vakter* og *lønnsadministrasjon.* Kravene er også prioritert, og kategorisert i ulike kategorier, i henhold til hvor viktig de er for systemet. De ulike kategoriene er "Må-krav" som er essensielle for at applikasjonen skal være nyttig for arbeidsgiver, "Bør-krav" og "Kan-krav". Den siste kategorien representerer funksjoner som er fine å ha, men som ikke nødvendigvis støtter kjernefunksjonene til applikasjonen. Den fullstendige listen over systemkrav ligger vedlagt i rapporten (se 8.4 systemkrav). Figur 5 under viser den prioriterte listen over krav.

MÅ - krav	
4.1	Daglig leder skal kunne logge inn med brukernavn og passord som gir tilganger som er forbeholdt administrator
1.1	Ansatte skal kunne logge inn med personlig brukernavn og passord.
2.2	Daglig leder og de ansatte skal kunne se vaktplan.
3.2	Daglig leder skal ved slutten av hver måned se en oversikt over månedens lønnsutbetaling.
3.3	Daglig leder skal kunne se detaljerte lønnsoversikt for hver enkelt ansatt (lønnsslipp).
4.2	Daglig leder skal kunne opprette ny bruker i systemet.
4.3	Daglig leder skal kunne endre alle opplysninger om en bruker.
2.1	Daglig leder skal kunne opprette/endre vakter
2.7	Ansatte skal kunne "sjekke inn og ut" av jobb.
2.11	Administrator skal kunne godkjenne alle som har sjekket inn og endre på dataen dersom det er behov.
2.12	Administrator skal ha egen side der han kan se hvem som er på jobb i nåværende tidspunkt.
BØR - krav	
1.2	Ansatte skal kunne melde seg ledig eller opptatt til vakter.
2.3	Daglig leder og de ansatte skal kunne se hvilke vakter som ikke er dekket de neste 10 dagene
2.10	Ansatte skal kunne se vakter, pause, mat og foreløpig estimert lønn som er registrert på dem, og kunne legge inn kommentarer på om opplysningene er riktig eller ikke.
5.1	Daglig leder skal kunne samle alle opplysninger som revisor trenger i et dokument som enkelt kan sendes videre.
2.8	Ansatte skal kunne registrere pause.
2.9	Ansatte skal kunne registrere mat.
2.4	Daglig leder skal få varsel om vakter som ikke er dekket de neste 5 dagene.
KAN - krav	

1.3	Ansatte skal kunne legge inn ønske om å bytte vakter, om de begge er ledig den vakten de ønsker å bytte til seg.
1.4	Ansatte skal kunne legge inn ønske om fri fra en vakt de allerede har registrert på seg.
1.5	Ansatte skal kunne legge inn tidsperioder hvor de er opptatt eller ønsker fri.
2.5	Daglig leder skal kunne se hvilke vakter de ansatte ønsker seg fri fra, og eventuelt godkjenne disse.
2.6	Ansatte skal bli varslet når ønsker om vakter, bytte av vakter og fri er blitt godkjent.
3.1	Daglig leder skal, når som helst, kunne se en estimert lønnsutbetalinger.

Figur 5, Systemkrav

3.2.4. Bruksmønster

Det ble utarbeidet en del bruksmønster på bakgrunn av systemkravene. Under er det et eksempel på bruksmønster for systemkravet 3.2 "Daglig leder skal ved slutten av hver måned se en oversikt over månedens lønnsutbetaling" (se 8.4 systemkrav under vedlegg). Vi lagde bruksmønster i fellesskap og bestemte på forhånd hva funksjonen skulle gjøre og hvorfor den var nødvendig. Vi har valgt å kun inkludere et bruksmønster i rapporten, se Figur 6. En komplett liste, i tillegg til flere eksempler (se vedlegg 8.5 Bruksmønster).

Navn	Se lønnsoversikt
Aktør	Administrator
Pre-betingelse	● Administrator må være pålogget på applikasjonen ● Dato for siste dag i lønns syklus må være forbi
Post-betingelse	● Endrede opplysninger blir lagret i databasen
Trigger	Administrator ønsker å se informasjon om lønnsutbetalinger
Normal hendelsesforløp	1. Administrator henter fram skjema for lønnsutbetaling. 2. Applikasjonen fyller feltene med data fra databasen. 3. Administrator kan endre / slette opplysninger. 4. Applikasjonen sjekker om alle påkrevde felter er fylt ut. 5. Applikasjonen lagrer endringene i databasen. 6. Applikasjonen viser lagrede endringer.
Variasjon	Administrator fyller ikke inn alle påkrevde felter

	1. Applikasjonen gir øyeblikkelig tilbakemelding til administrator om hvilke felter som mangler informasjon.
Relatert informasjon	I henhold til systemkrav 3.2

Figur 6, Bruksmønster

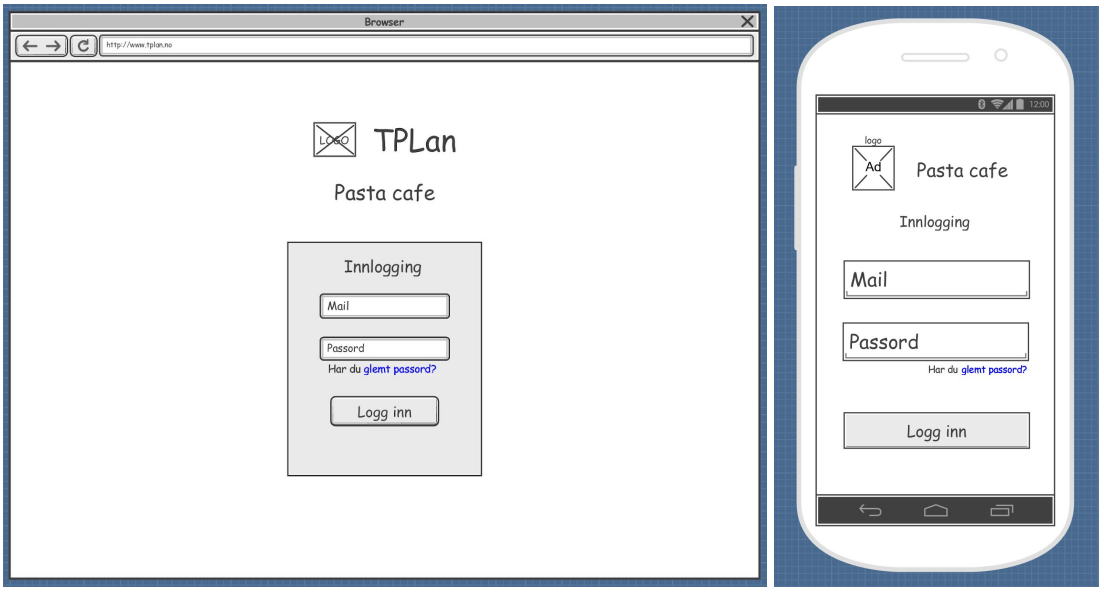
3.3. Designfasen

I denne delen presenterer vi hvordan vi har kommet frem til det visuelle designet. I tillegg viser vi hvordan databasen er bygget opp for å støtte applikasjonen. Til slutt forklarer vi hvilke programmeringsspråk og verktøy vi har brukt for å utvikle applikasjonen

3.3.1. Wireframes

Når vi jobbet med å lage wireframes satt vi sammen for at alle kunne komme med innspill, og for å bli enig om en visuell profil. Denne prosessen gjorde at vi fikk diskutert flere aspekter ved applikasjonen enn det visuelle designet. Vi brukte wireframes for å hjelpe oss med å visualisere innhold, navigering, plassering av innhold, skjemaer og knapper som skulle utføres i webapplikasjonen. Vi lagde wireframes ved å bruke verktøyet Visual Paradigm. Vi satt sammen og utførte hver wireframes i lag, for å diskutere designet av applikasjonen. Grunnen vi lagde wireframes i lag, var for å komme til en enighet om hvordan applikasjonens utseende skulle være og i forhold til universell utforming, før vi skulle gå videre til neste Sprint som var implementering, som bestod av å kode.

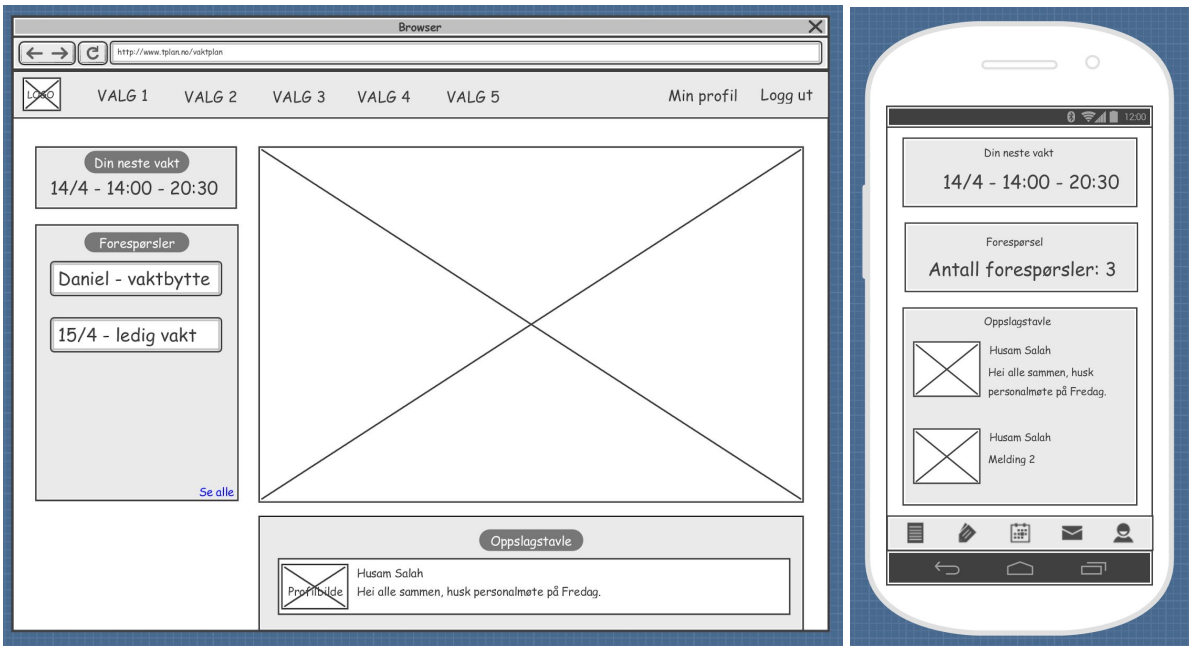
Vi har brukt standarder innenfor webutvikling for å plassere sentrale elementer hvor brukere kjenner seg igjen og finner det de leter etter. Typiske eksempler på slike normer er at meny og søkemotor er plassert øverst på siden, og profilbilde med drop-down meny for utlogging øverst til høyre (Crestodina, u.å.).



Figur 7, Pålogging-Wireframe

Bildet ovenfor viser at det første ansatte møter i applikasjonen er påloggings-skjemaet vårt. Der kreves at de fyller inn både brukernavn og passord for å logge inn før de kan gå videre. I tillegg får brukere mulighet til å få tilsendt nytt passord om brukeren har glemt det gamle passordet sitt.

Vi har også fokusert på at applikasjonen skal fungere både på mobil og PC enheter slik at den fungere uavhengig av hvilken enhet brukeren benytter seg av. Nedenfor vises et eksempel på wireframes som ble laget til applikasjonen. Alle wireframes ligger som vedlegg til rapporten (se 8.6 Wireframes).

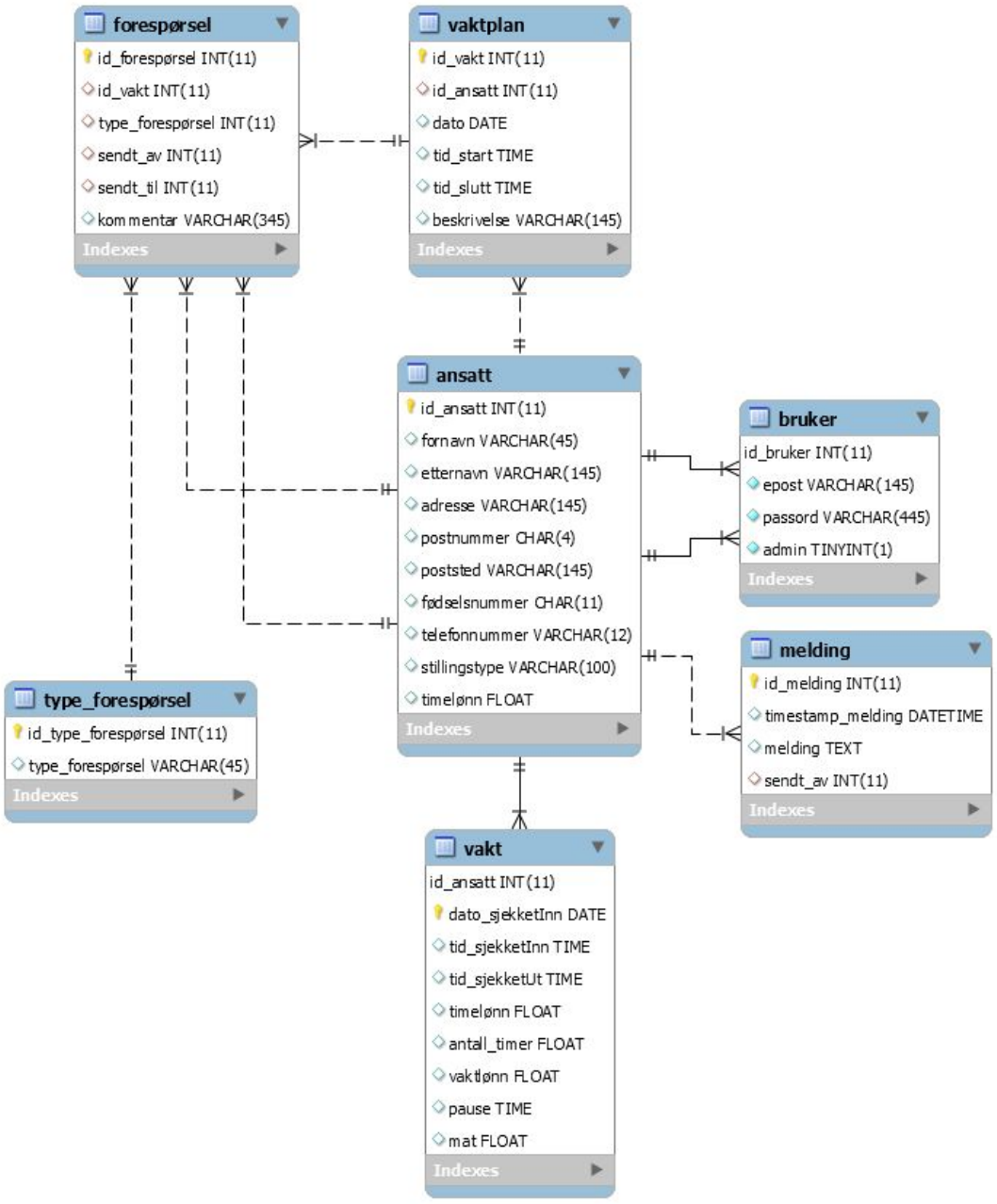


Figur 8, Hovedsiden- Wireframe

Bilde ovenfor viser wireframe for hovedsiden, i både stor og liten skjerm. Wireframes gjorde det enkel for å programmere systemkravene.

3.3.2. Databasemodell

Ved bruk av databasemodell kan man visuelt vise fram strukturen til en database. (Kristoffersen, 2016, s. 4). Under vises den første versjonen av databasen vår. Vi regner med å måtte endre denne da implementering av systemkrav vil avdekke eventuelle feil og mangler databasestrukturen har. Den endelige løsningen for databasen vår vil bli beskrevet under punkt 4.2 i systemdokumentasjonen.



Figur 9, Databasemodell

3.3.3. Valg av utviklingsspråk og verktøy

Vi har brukt forskjellige teknologier for å utvikle denne applikasjonen, her skal vi gå inn mer på dybden av hver eneste teknologi. Her drøfter vi om alternativer og hvorfor vi valgte å jobbe med disse.

Vi har utviklet vår applikasjon i HTML og CSS fordi vi har god erfaring med språket og gruppemedlemmene har brukt det aktivt siden studiestart. Noen av fordelen med dette HTML er at den er stabilt, og, rammeverket til språket er allerede definert og utvikleren kan fokusere på å være kreativ. Ulemper er blant annet at utvikleren må være kreativ og plassere elementene på siden på egenhånd. Her gjelder det å plassere elementene på siden på en ryddig måte, slik at det ikke skaper kaos. Oftest vil elementene plasseres likt som andre nettsider, men dette er en fordel når brukere er allerede kjent hvor de "normale" elementene er plassert. For eksempel at menyen er plassert på toppen på en laptop plattform.

Andre alternativer er for eksempel JavaScript, som er mer avansert programmeringsspråk enn HTML/CSS. JavaScript er populært språk og blir brukt i mange av dagens nettsider. Ofte blir JavaScript brukt som bekreftelses-bokser, for eksempel der man fyller ut et skjema med informasjon, også dukker det opp et boks der man må klikker på enten "OK" eller "Avbryt" for å fortsette (Kolowich, 2018). Vi valgte å ikke bruke JavaScript, fordi ingen av gruppemedlemmene hadde erfaring med det språket og ville spare tid på å lage en god applikasjon.

Webapplikasjonen er avhengig av en robust database. Databasen brukes for å hente, endre, lagre, oppdatere og lese dataene som er basert på forespørsler fra brukerne. I tillegg hjelper databasen med å vise oppdaterte informasjon til brukerne. Vi har valgt å bruke PHP og MySQL Workbench, fordi de samarbeider godt med databasen og HTML/CSS språket. Med PHP og SQL spørringer kan vi begrense informasjon for brukere som har ulike tillatelser basert på deres jobbrolle, vise feilmeldinger dersom det går noe galt med sending av data til databasen og mer.

Vi valgte å bruke PHP siden webapplikasjonen benytter databaseverktøyet MySQL, og JavaScript kan ikke kobles til databasen vår. JavaScript er språk på klientsiden som kjøres i

nettleseren. Mens PHP er på serversiden, og tolker koden for å sende resultatet til nettleseren (Khillar, 2018).

De teknologiene vi valgte å benytte oss av, gjorde at vi kunne få ut mest mulig funksjonalitet i webapplikasjonen. For å designe utseende til både på PC-skjerm og mobiltelefoner benyttet vi Bootstrap. Bootstrap er et bibliotek med alle elementene som HTML/CSS tilbyr, forskjellen er at elementene er konfigurert på forhånd og skaper et behagelig utseende for brukerne. Hvis utvikleren er ikke fornøyd med fargen eller designet til en tabell for eksempel, har utvikleren muligheten til endre på de. Med å bruke Bootstrap sparer man tid og er et responsiv verktøy. Siden Bootstrap inkluderer responsiv verktøy vil elementene være designet til å forme seg etter enheten som blir brukt, en knapp vil da endre sin bredde avhengig av skjermstørrelse. Vi kan velge hvilke elementer vi vil bruke fra biblioteket til Bootstrap og legge det til i webapplikasjonen vår.

For å kunne jobbe med alle disse teknologiene har vi brukt ulike verktøy. Det viktigste verktøyet har vært Visual Studio Code (VSC), og den ble brukt til å kode selve webapplikasjonen og kan håndtere mesteparten av programmeringsspråkene som finnes. Den har noe som heter IntelliSense, og den forsøker å gjette hva utvikleren vil skrive for å fullføre koden man jobber med og hjelper med å redusere feil i koden. Den har innebygd Git støtte, som gjør det lettere for oss å bruke GitLab for versjonskontroll og kunne synkronisere koden vår i og med at vi er flere utviklere i gruppen. VSC har en enkel feilsøking (debugging) verktøy og du kan når som helst legge til andre utvidelser (Microsoft, u.å.). Vi kunne valgt å bruke andre verktøy som Atom, Brackets eller Notepad++ som også får jobben gjort, men VSC effektiviserer utviklingen med tanke på alle små utvidelsene som den har.

Vi brukte domeneshop som for webhotell og domene. Ved hjelp av Domeneshop, kunne vi alle være koblet til serveren hvor som helst og når som helst. Domeneshop tilbyr flere tilleggstjenester som vi har behov for, blant annet en database. Her kjøpte vi et domene, og oppdragsgiveren kan ta over dette eierskapet senere dersom han ønsker å beholde det. Alternativer var å benytte seg av skolens server men da var vi bundet til å være på skolen for

å være koblet mot serveren. Dermed ble det lettere for oss å jobbe med webapplikasjonen vår nå den lå på Domeneshop.

3.4. Implementering

Utviklingen av applikasjonen er delt opp i Sprinter i henhold til rammeverket Scrum. Vi realiserer systemkravene i henhold til den prioriterte listen (se vedlegg 8.4 Systemkrav) slik at de viktigste funksjonene blir gjennomført først. Noen av systemkravene er satt sammen for å gjennomføre i samme sprint, av samme gruppemedlem, fordi det er tidsbesparende da koden eller logikken kan være lik.

Vi holdt oss til tidligere nevnt arbeidsplan med 18 timers arbeidsuke per gruppemedlem. Derfor regner vi som en tommelfingerregel 36 timer per 2 ukers Sprint. Vi skriver i etterkant av sprinten om antall timer ble brukt og vil kommentere hvis det tok vesentlig lenger eller kortere tid å innfri systemkravene. Vi benyttet GitLab slik at alle gruppemedlemmene fikk samtlige oppdateringene og alle hadde tilgang til oppdatert kode. I starten møtte noen av oss på tekniske problemer, da VSC ville ikke koble seg mot GitLab. Årsaken til dette, var at det var krypterings feil på GitLab brukeren, og vi løste dette problemet med å følge brukerveiledningen fra GitLab. Vi så da fordelen ved å bruker samme utviklingsverktøy fordi det var enklere å problemløse, og å hjelpe hverandre.

Noen viktige systemkrav hadde vi ikke kartlagt fra starten, og ble lagt til underveis. For eksempel 2.1 Daglig leder skal kunne opprette/endre vakter. Denne funksjonen gikk i glemmeboken, fordi vi tenkte at vaktene kommer fra databasen, uten at vi tenke gjennom hvordan de skulle opprettes der. Dette problemet ble løst i Sprint 2.

Vi har valgt å forkorte tittelen på noen systemkrav når vi har brukt de som underoverskrift for at de ikke skal bli for lange. Nummeret til systemkravet er det samme slik at de ikke skal være tvil om hvilket systemkrav det er snakk om.

Dokumentasjonen av arbeidet i sprintene består blant annet av skjermbilder av applikasjonen. Skjermbildene inneholder testdata som ikke nødvendigvis er reelle.

3.4.1. Sprint 1

Denne første sprinten markerer starten på selve programmeringsjobben. Vi har fordelt systemkrav etter den prioriterte listen. Oppgavene er fordelt slik at vi regner med å bruke ca 36 timer hver iløpet av en sprint som varer i to uker. Gruppemedlemmene kom med ønsker om hva de ville jobbe med, og vi ble enig om fordelingen.

Start: 5.mars Varighet: 2 uker

Systemkrav	Ansvarlig	Timer estimert	Timer brukt
1.1 Ansatte skal kunne logge inn med personlig brukernavn og passord 4.1 Daglig leder skal kunne logge inn med brukernavn og passord som gir tilganger som er forbeholdt administrator	Majd	36 timer	29
4.2 Daglig leder skal kunne opprette ny bruker i systemet. 4.3 Daglig leder skal kunne endre alle opplysninger om en bruker.	Daniel	36 timer	20
2.1 Daglig leder og de ansatte skal kunne se vaktplan.	Roza	36 timer	38
3.2 Daglig leder skal ved slutten av hver måned se en oversikt over månedens lønnsutbetaling	Elise	36 timer	30

Figur 10, Sprint 1

Systemkrav 1.1 og 4.1

For å kunne programmere funksjonalitetene for innlogging måtte vi lage testbrukere i databasen. Webapplikasjonen bruker inn-data hentet fra innloggings-skjema og sammenligner data fra databasen. Dersom brukeren eksisterer i databasen og passordet er riktig sjekker applikasjonen om brukeren er administrator eller ikke, og henter tilhørende forside. Vi opplevde problemer underveis da systemet hadde vanskeligheter med å skille

brukertypene. Vi løste problemet med å systematisk feilsøke koden og fant ut at en variabel var definert feil.

Systemkrav 4.2 Daglig leder skal kunne opprette ny bruker i systemet

Vi lagde en side for å opprette nye brukere, det er kun de med administrasjonstilgang som kan utføre denne funksjonen. Det første som skjer når en administrator prøver å opprette en ny bruker er at det sjekkes om brukeren finnes i databasen fra før. Dersom brukeren, eller opplysninger som blant annet er e-post eller telefonnummer eksisterer, vil applikasjonen gi feilmelding om hva som er galt. Hvis alle opplysningene er riktig fylt ut vil applikasjonen opprette en ny bruker.

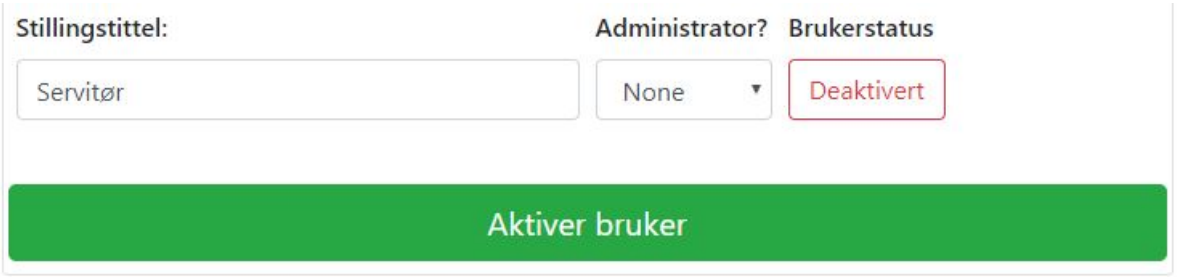
Dette systemkravet tok lenger tid enn antatt fordi det var utfordrende å finne ut om brukeren eksisterte. Grunnen til dette var at vi ønsker å sjekke flere opplysninger om brukeren individuelt. På denne måten unngår vi at administrator kan lage flere brukere på samme person ved at én opplysning skiller seg fra en eksisterende bruker.

Vi hadde regnet med at dette skulle ta ca. en uke for å kunne fullføre all funksjonalitet og brukervennligheten på plass, men vi overskred tiden med flere timer merarbeid. Vi fikk en del feil ved registrering av brukere og sende det inn til *ansatt* og *bruker* tabellen i databasen. Vi fikk også en del feil på funksjoner som skal vise feilmeldinger dersom data eksisterer, men det fikk vi løst på. Det som mangler er passordet, fordi vi ønsker at sikkerheten skal være sterk og da er det nødvendig å ha passord med god styrke. Vi prøvde å få til at dersom passordet blir skrevet inn , så skal det inneholde minst ett stort forbokstav, små bokstaver, tall og godkjente tegn om nødvendig. Men vi får feilmeldinger på det og det lar seg ikke å gjøre, men dette er noe vi kan jobbe med senere dersom tiden lar seg tilstrekke. Men vi har brukt såkalt salting og hashing metode for å sørge for at passordet som blir lagret i databasen består av forskjellige små og stor forbokstav og tall. Dette gjorde vi slik at ingen har tilgang til å kunne se passordet til brukeren etter opprettelse av bruker.

4.3 Daglig leder skal kunne endre alle opplysninger om en bruker

Her hentes det ut en liste over alle brukere som er lagret i systemet og administrator kan da velge en ansatt som han vil endre opplysninger om. Vi har gjort det slik at administrator kan deaktivere eller aktivere brukere. Brukere skal kunne deaktiveres og ikke slettes, fordi dataene som er lagret kan være aktuell for arbeidsgiver selv om en ansatt har sluttet. Deaktiverte brukere kan ikke endres på med mindre de blir aktivert igjen. Administrator kan endre på all informasjon om en ansatt med unntak av passord. Som vist nedenfor prøver vi å gjøre applikasjonen med brukervennlig ved å kun vise relevante knapper. En bruker som er aktiv kan endres eller deaktiveres og brukere som er deaktivert kan kun aktiveres.

Figur 11, Endre brukere



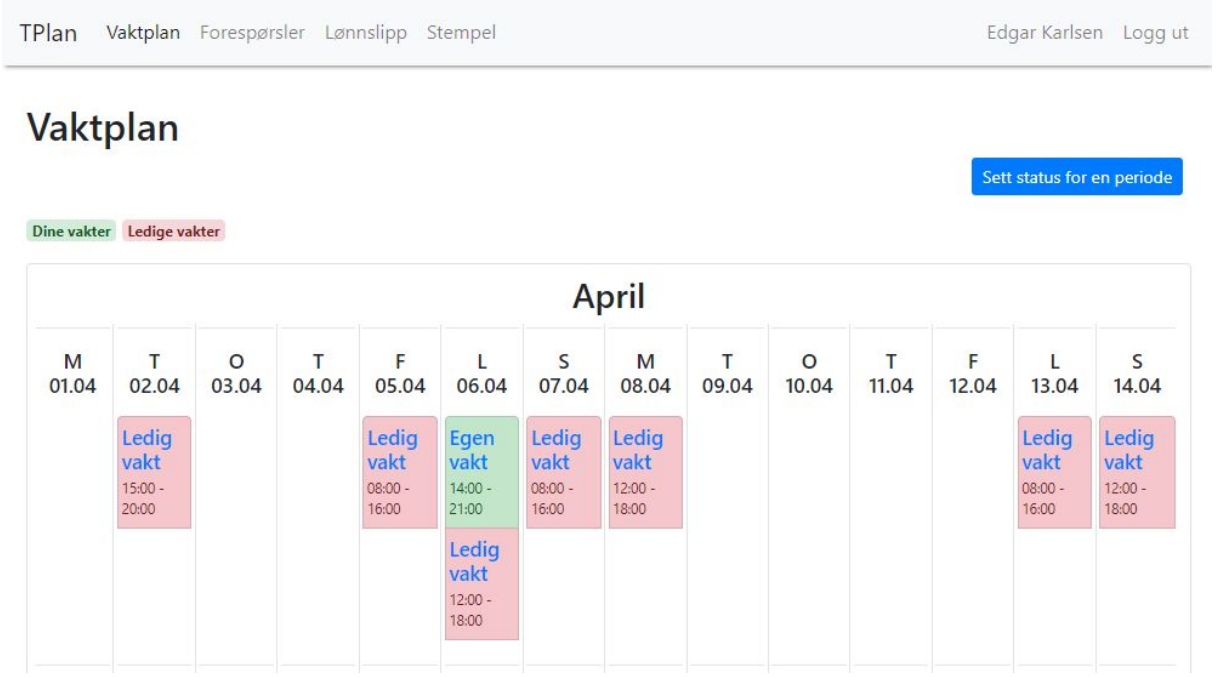
Figur 12, Aktivere brukere

2.1 Daglig leder og de ansatte skal kunne se vaktplan

Dette systemkravet ble tidlig delt i to da vi ønsket å vise en vaktplan for inneværende uke "Ukeplan", for å ha på forsiden. Denne viser alle vaktene og hvem som jobber denne uken.

Figur 13, Ukeplan forside

Del to som er er vist under, var en vaktplan på en egen side for flere måneder fram i tid der man kun ser egne, og ledige vakter. Vaktplanen viser fra første dag av inneværende måned og til sammen seks måneder fram i tid. Dette er for å gi ansatte en oversikt som man kan bruke til å planlegge fri og ferie.



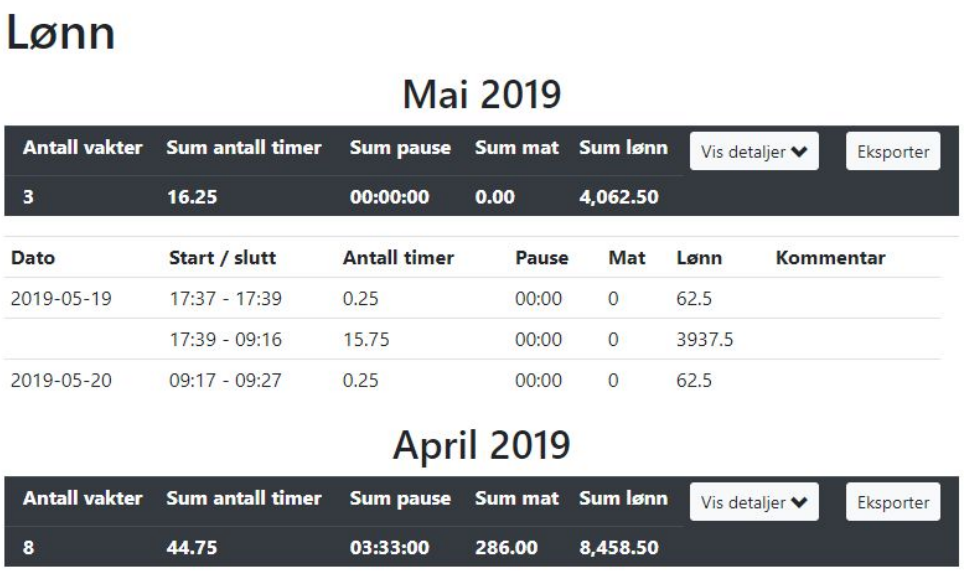
Figur 14, Vaktplan

Vi startet med å sjekke om vi får data fra databasen, det gjorde vi med å legge inn testdata for de ulike ansatte, og sjekket om de dukket opp på applikasjonen. Videre stylet vi vaktplanen med å lage en ryddig oversikt over de ulike månedene og ukene. For vaktplanen valgte vi å vise 14 dager på hver linje, og et tydelig skille mellom månedene, for at brukeren lettere skal få oversikt. I tillegg brukte vi ulike farger som skulle definere ledige vakter og vaktene til den spesifikke ansatte som var logget inn. Videre vises dagene med kun forbokstav, vi mente dette var tydelig nok.

Vi brukte lenger tid enn antatt på dette systemkravet, da det var utfordrende å lage en vaktplan med kalender for flere måneder som forholdt seg til ulikt antall dager i en måned, og hvor månedene starter på ulik ukedag. Vi gjorde dette kanskje ekstra vanskelig for oss selv når vi valgte å gjenbruke kode fra ukeplan på forsiden som viser nåværende, neste og forrige uke. Logikken er annerledes når ukene alltid starter på mandag, og du ikke trenger å forholde deg til antall dager i en måned. Gjenbruk av kode er ikke alltid like effektivt da det kan være logisk vanskeligere å endre kode enn å skrive koden fra bunnen av.

3.2 Daglig leder skal se en oversikt over månedens lønnsutbetaling

Vi ønsket at daglig leder skulle kunne se oversikt over månedens lønnsutbetalinger. Vi valgte å først gi et oversiktsbilde hvor daglig leder ser summerte utgifter for hver måned. Han kan velge å se detaljert informasjon, sortert på dato og vakt. Figur 24 under viser hvor detaljene til mai måned er åpnet.



Figur 24, Lønnsutbetaling

Knappen "Eksporter" igangsetter nedlastning av en Excel-fil med informasjonen på siden.

3.4.2. Sprint 2

Etter en godt gjennomført Sprint 1, var vi motiverte til å fortsette nedover listen over systemkrav i Sprint 2. Den første sprinten ga oss erfaringen om hvordan vi kan kombinere systemkrav som har lignende logikk. Derfor har flere gruppemedlemmer fått tildelt flere funksjoner.

Start: 19.mars Varighet: 2 uker

Systemkrav	Ansvarlig	Timer estimert	Timer brukt
2.7 Ansatte skal kunne "sjekke inn og ut" av jobb. 2.11 Administrator skal kunne godkjenne alle som har sjekket inn og endre på dataen dersom det er behov. 2.12 Administrator skal kunne se hvem som er på jobb.	Majd	36	40

2. 1 Daglig leder skal kunne opprette/endre vakter.	Daniel	36	36
3.3 Daglig leder skal kunne se detaljerte lønnsoversikt for hver enkelt ansatt (lønnsslipp).	Roza	36	16
1.2 Ansatte skal kunne melde seg ledig eller opptatt til vakter. 1.3 Ansatte skal kunne legge inn ønske om å bytte vakter, om de begge er ledig den vakten de ønsker å bytte til seg. 1.4 Ansatte skal kunne legge inn ønske om fri fra en vakt de allerede har registrert på seg.	Elise	36	38

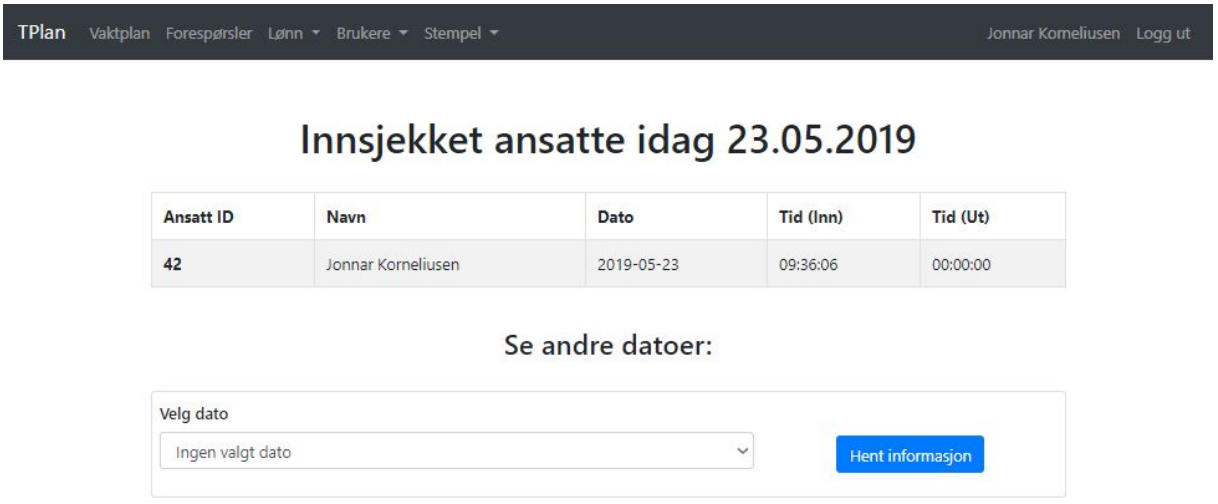
Figur 26, Sprint 2

2.7 Ansatte skal kunne "sjekke inn og ut" av jobb

Denne funksjonen er essensiell for å kunne beregne lønnen til ansatte, denne skal brukes hver dag av alle som kommer på jobb. Den bidrar også til å oppfylle systemkrav 2.12 "Administrator til enhver tid skal kunne se hvem som er på jobb". Igjen har vi valgt å fjerne knapper og funksjoner som ikke er relevante, derfor funker den slik at hvis en ansatt ikke har sjekket inn så vises det kun en "Sjekk inn" knappen. Dersom brukeren allerede har sjekket inn, får man andre alternativer som er vist under. Med dette kan ansatte registrere kostnader rundt mat og pauser, deretter vil applikasjonen regne ut lønn avhengig av ansattes timelønn, og lagre dette i databasen. Alle vakter som blir lagret må godkjennes av administrator.

Figur 27, Utsjekk

2.12 Administrator skal kunne se hvem som er på jobb



Figur 28, Innsjekking

Ifølge oppdragsgiver krever Arbeidstilsynet at han har kontroll på hvem som er på jobb til enhver tid og at det blir dokumentert. Denne siden henter ut informasjon om alle ansatte som er eller har vært pålogget per dags dato. Vi har også gjort at administrator kan velge å se liste over andre datoer dersom en ønsker dette.

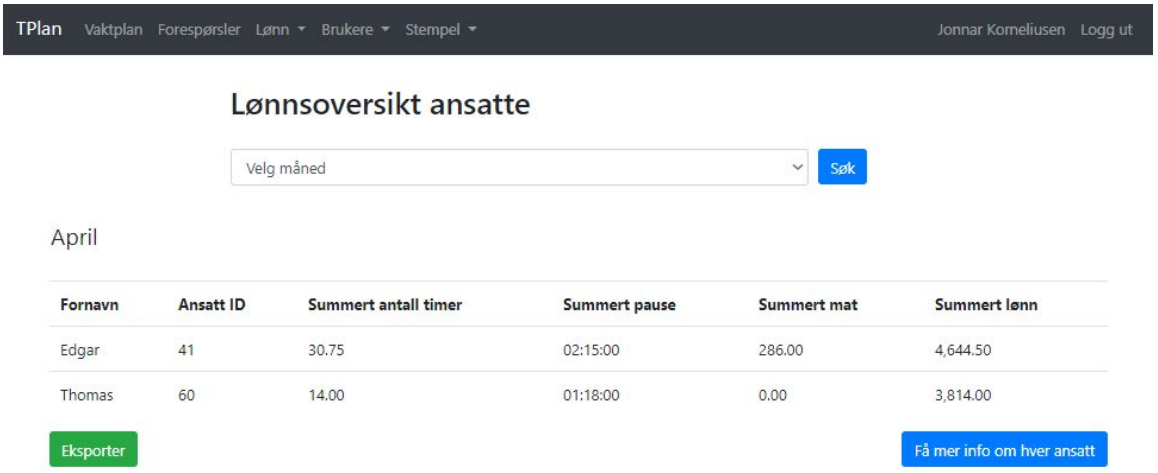
2.1 Daglig leder skal kunne opprette/endre vakter

For at daglig leder skal opprette vakter til sine ansatte, så er det viktig med et skjema som registrerer inn vakter. Skjemaet legger frem viktige felter som dato for vakten, klokkeslett for start og endt vakt, en liste med registrerte ansatte og beskrivelse om vakten og til slutt en opprett knapp registrerer inn vaktene til databasen. Dersom daglig leder har gjort et glipp eller må endre på en vakt, har vi laget en side hvor han kan hente frem vakta. Han velger dato for gjeldende vakt deretter hentes informasjon fra databasen, og får da to valgknapper dersom vakten eksisterer, "endre" og "slett" knapp. Når "endre" knappen trykkes, dukker det opp et skjema med felter som kan endres på og til slutt blir oppdatert. Dersom daglig leder trykker på slett knappen, slettes vakten fra databasen.

3.3 Daglig leder skal kunne se detaljerte lønnsoversikt for hver enkelt ansatt

Daglig leder kan se ansattes lønnsslipp for hver måned. Han velger hvilke måned i søkemotoren deretter vises data fra MySQL databasen. Dataene vises etter hvem som er logget inn.Ved å velge måned får han opp en oversikt der lønnsslippene til de ansatte er summert for valgt måned. Knappen "Eksporter" laster ned denne oversikten til en Excel-fil.

Ved å trykke på "Få mer info om hver ansatt" blir man sendt videre til en ny side med lønnsslippen til hver enkelt ansatt.



Figur 29, Lønnsoversikt ansatte

På den nye siden, lønnsslipper, velger man måned og ansatt for å se lønnsslippen. Figur 30 viser resultater av valget April og brukeren Edgar.

Lønnsoversikt Edgar for April

Dato	Timer	Pause	Mat	Timelønn	Tilleggstimer	Tilleggstimer pris	Vaktlønn
26.04	8.5	00:30	156	145	4	67	1069.5
27.04	4.75	00:30	60	250	0	0	1127.5
24.04	3.25	00:00	35	145	0	0	436.25
17.04	5.25	00:45	35	145	0	0	726.25
15.04	9	00:30	0	250	0		1285
Summert	**Timer: 30.75**	**Pause: 02:15:00**	**Mat: 286.00**		**Tilleggstimer: 4**	**Tilleggstimer pris: 67**	**Lønn: 4,644.50**

Figur 30, Lønnsslipp

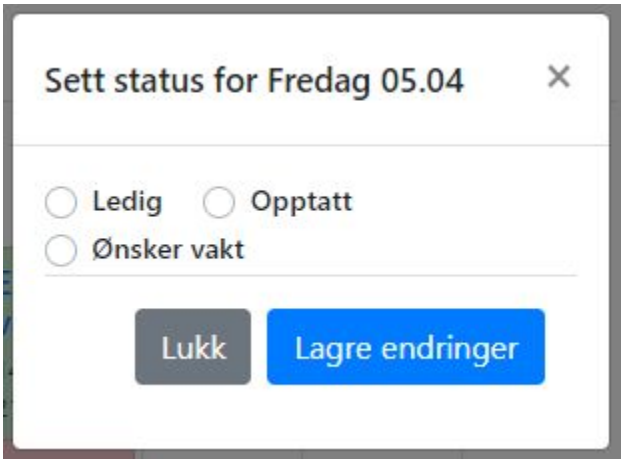
1.2 Ansatte skal kunne melde seg ledig eller opptatt til vakter.

Før vi startet arbeidet med dette systemkravet oppdaget vi at databasen hadde ikke tabeller som støtter denne funksjonen. Derfor la vi til en tabell over statuser og en koblingstabell for å knytte vakter fra vaktplan med ansatt og status. På den måten kan vi se hvilken status hver ansatt har satt for en bestemt vakt. Endringene i databasen gjør at vi må gjøre endringer i innholdet i tabellen *type_forespørsel* som nå inneholder blant annet "ledig vakt". Denne typen forespørsel er overflødig da vi sier at en vakt er ledig når den ikke er gitt til en ansatt, med andre ord er feltet ansatt i tabellen *vaktplan* uten innhold. Figur 31 viser vakter som har ansatt tilegnet seg, med id_ansatt, og vakter som er ledige med id_ansatt satt til "NULL".

id_vakt	id_ansatt	dato	tid_start	tid_slutt	beskrivelse
23	6	2019-03-05	12:00:00	18:00:00	Kokk
24	7	2019-03-06	14:00:00	21:00:00	Stengevakt
25	NULL	2019-03-07	08:00:00	16:00:00	Åpnevakt
26	NULL	2019-03-08	12:00:00	18:00:00	Kokk

Figur 31, Databasetabellen vaktplan

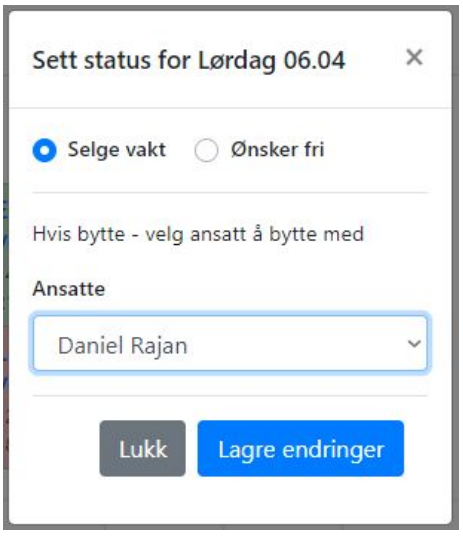
Dermed fjernes "ledig vakt" som en type forespørsel. Nedenfor vises hvordan vi løste systemkravet. Via vaktplan kan ansatte klikke på ledige vakter og dermed åpnes en boks hvor man kan huke av for den statusen de ønsker å gi vakten. Statusene hentes direkte fra databasen og er derfor adaptive til endringer. Når en bruker velger å lagre statusen oppdateres databasen og nettsiden hentes på nytt for å vise status for vakten i vaktplan.



Figur 32, Valg for vakter

Figur 33, Ledige vakter

Som en naturlig forlengelse av systemkravet la vi til mulighet til å be om fri, eller å bytte/selge egne vakter. Her bruker vi databasetabell *type_forespørsel* og forespørsel til å sende forespørsel som daglig leder / admin, og den som man eventuelt vil bytte vakt med må godkjenne. Når man trykker på egen vakt i vaktplanen får man opp en boks hvor man kan "selge" vakten og velge hvem man ønsker å selge den til, eller eventuell be om fri. For å se et eksempel på dette, se figur 34.



Figur 34, Sprettoppvindu forespørsler

3.4.3. Sprint 3

Etter Sprint 1 og Sprint 2 begynner applikasjonen å ta form, og i Sprint 3 tar vi for oss systemkrav som har lavere prioritert. Det er likevel funksjoner som vi føler det er viktig å få på plass for at applikasjonen skal gi oppdragsgiver verdi.

Start: 2.april Varighet: 2 uker

Systemkrav	Ansvarlig	Timer estimert	Timer brukt
5.1 Daglig leder skal kunne samle alle opplysninger som revisor trenger i et dokument som enkelt kan sendes videre.	Majd	36	32
2.5 Daglig leder skal kunne se hvilke vakter de ansatte ønsker seg fri fra, og eventuelt godkjenne disse.	Daniel	36	28
2.4 Daglig leder skal få varsel om vakter som ikke er dekket de neste 5 dagene.	Roza	36	28
1.5 Ansatte skal kunne legge inn tidsperioder hvor de er opptatt eller ønsker fri.	Elise	36	30

Figur 35, Sprint 3

5.1 Daglig leder skal kunne samle alle opplysninger som revisor trenger i et dokument som enkelt kan sendes videre.

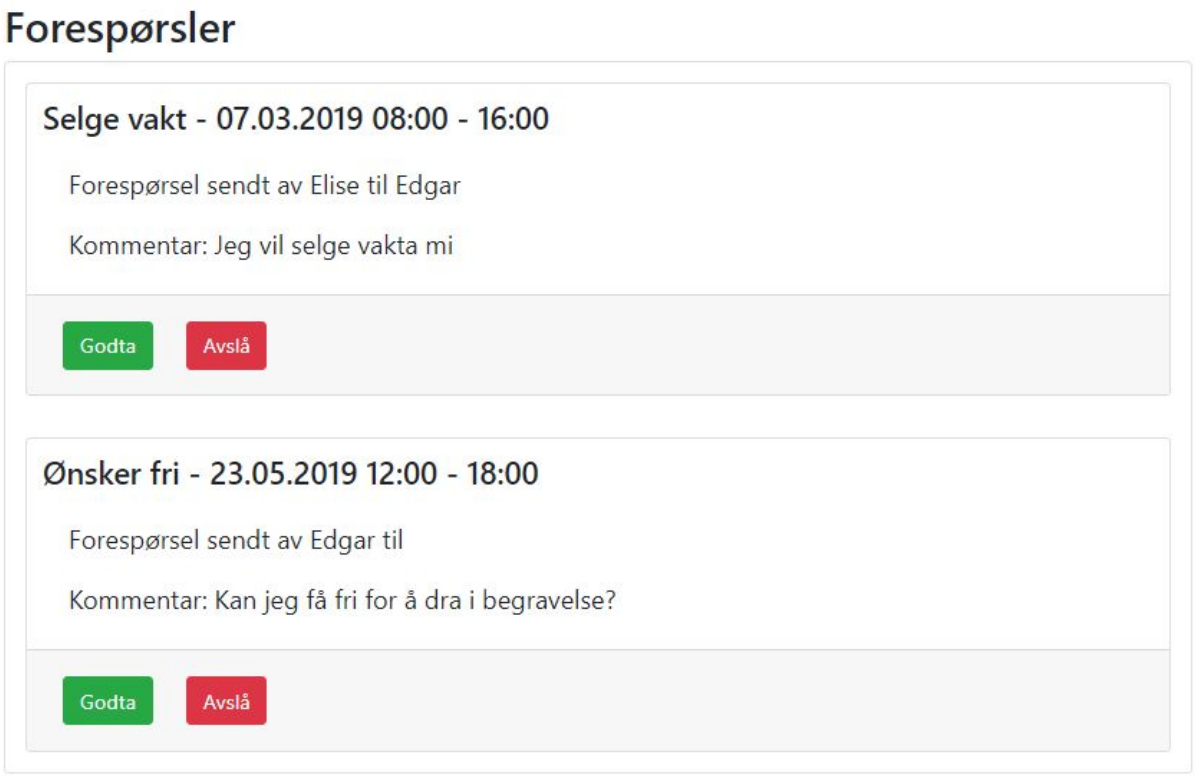
Dette systemkravet har tatt lenger tid enn antatt, grunnen til det er at det er tre skritt for å kunne eksportere informasjonen til Excel. Først må vi kunne hente ut riktig data fra databasen og sikre at det er riktig data, det tok litt lenger tid å dobbeltsjekke dataen fordi den hentes fra flere forskjellige tabeller fra databasen og med tanke på at informasjonen skal brukes til lønnsutbetaling er det utrolig viktig at det er ingen feil. Andre steg er å kunne vise dataene i en tabell, det er ikke en vanskelig oppgave men tidkrevende. Siste steget var å sende dataene til Excel og starte nedlastingen hos brukeren, her prøvde vi et par metoder for å kun få inn tabellen i Excel og ingenting annet. Mange av fremgangsmåtene vi prøvde i starten var å eksportere hele applikasjonen, i motsetning til kun tabellen dette var vår største utfordring når det gjaldt eksportering. Til slutt løste vi problemet med å åpne tabellen med dataen i en blank side, deretter starte nedlastingen og sende brukeren tilbake til riktig side.

	A	B	C	D	E	F	G	H	I	J	K
1	**Ansatt ID**	**Fullt navn**	**Fastlønn**	**Timespris**	**Antall timer**	**Tilleggstimer**	**Tilleggstimer pris**	**Kost**	**Summert lønn**	**Forskudd**	**Diverse**
2	41	Edgar Karlsen		250	feb.00	0.00	0.00	60.00	1127.5		
3	42	Jonnar Korneliusen		145	22.00	4.00	67.00	226.00	2069.5		
4											
5											

Figur 36, Lønningsdetaljer i Excel

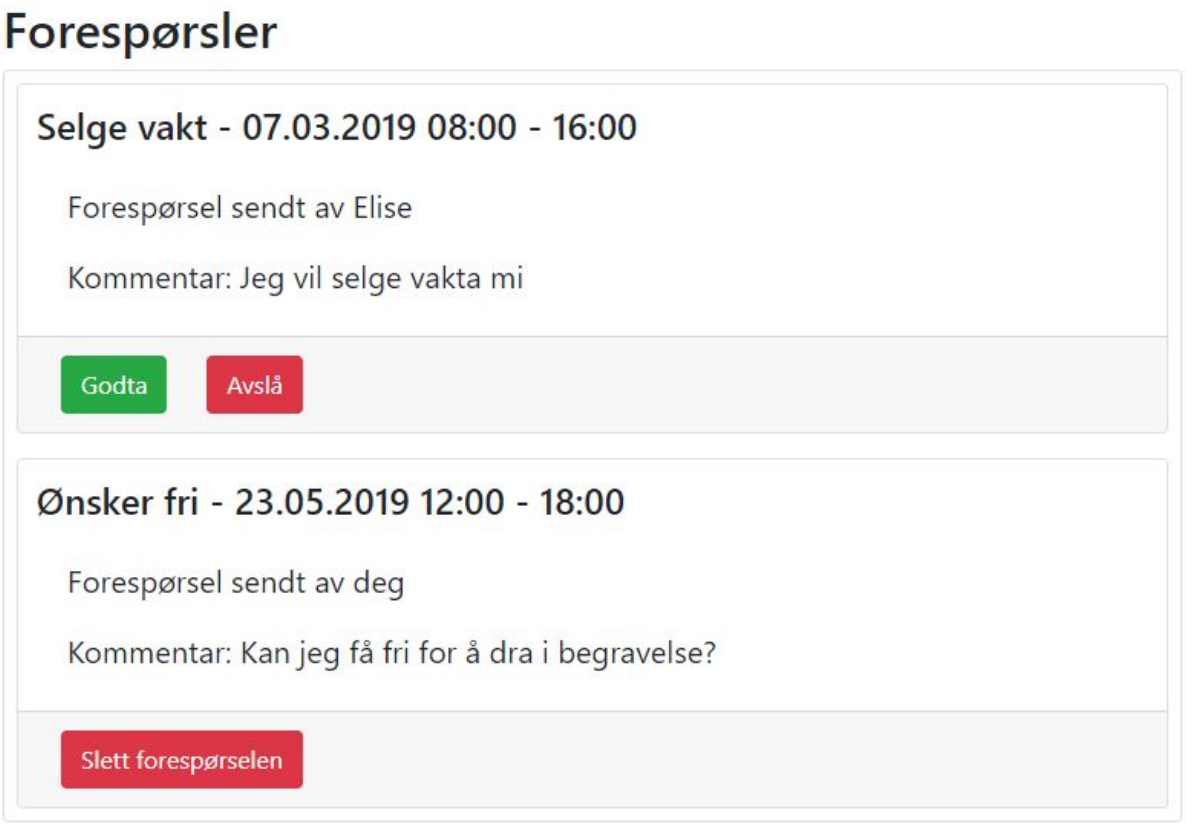
2.5 Daglig leder skal kunne se forespørsler og eventuelt godkjenne disse

I Sprint 3 utviklet vi en funksjon hvor de ansatte kan sende forespørsel om fri fra en vakt, eller å selge vakten sin til en annen ansatt. Denne funksjonen ble til som en naturlig forlengelse av systemkrav 1.2 hvor de ansatte skal kunne sette status på ledige vakter. Dette systemkravet har derfor blitt utvidet siden vi formulerte den i analysefasen. Nå tenker vi at det er naturlig med en side hvor daglig leder får en oversikt over forespørsler, både om fri og om å selge vakt, slik at han kan godkjenne eller avslå de. Figur 37 under viser hvordan forespørsler blir presentert for administrator.



Figur 37, Forespørsler administrator

I arbeidet med dette systemkravet så vi at en annet, lignende, systemkrav for ansatte kunne implementeres ved gjenbruk av den koden som ble skrevet for administrator. Ved å gjenbruke koden, men å kun vise forespørsler som er sendt av, eller til, den spesifikke ansatte kunne vi lage en side med forespørsel til ansatte også. Systemkravet 2.6 "Ansatte skal bli varslet når ønsker om vakter, bytte av vakter og fri er blitt godkjent". Blir ikke oppfylt slik tanken opprinnelig var, men det er et KAN-krav og vi mener at denne løsningen er bedre enn ingenting. Figur 38 under viser hvordan forespørsler ser ut for de ansatte. Her kan de velge å godta eller avslå forespørsler som er sendt til de eller slette forespørsler som de har sendt.

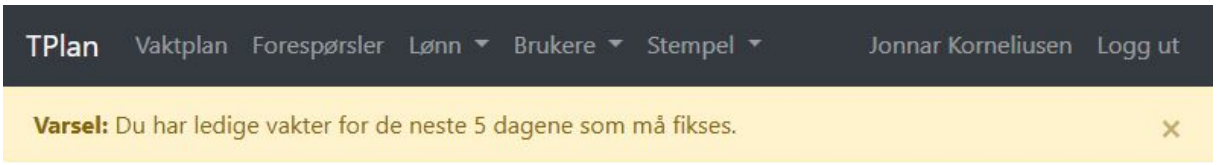


Figur 38, Forespørsler ansatt

Vi valgte også å inkludere en oppsummering av de forespørslene ansatte har på forsiden, med link til siden med forespørsler.

2.4 Daglig leder skal få varsel om vakter som ikke er dekket

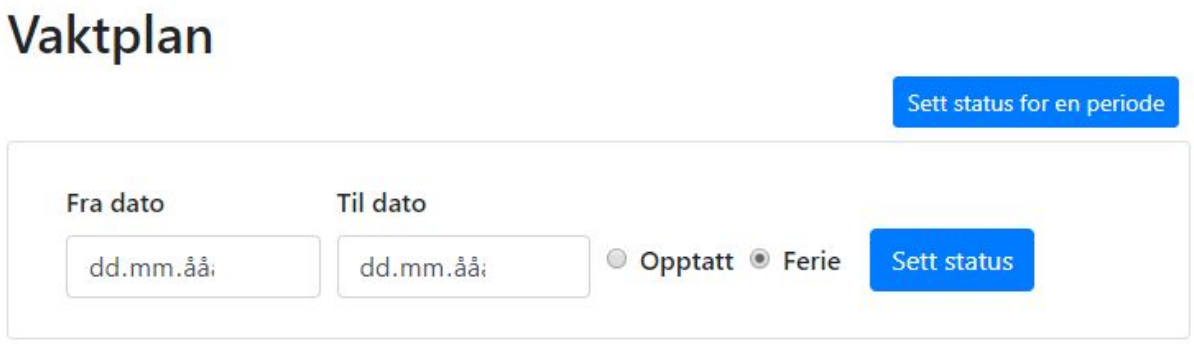
Denne funksjonen skal vise varsel på siden til administratoren om at vakter innen fem dager som ikke er dekket. Vi valgte fem dager, fordi vi mente da er det krise og i tillegg får han et par dager til å løse dette problemet. Bilde under viser at et gult felt har dukket opp under navbaren, som signaliserer at at det er fem vakter som ikke er dekket. I tillegg har brukeren mulighet til å trykke på den kryssen (X) til høyre for å fjerne varselen. Hvis det er ingen ledige vakter, da vises det ingen varsel.



Figur 39, Varsling om udekte vakter

1.5 Ansatte skal kunne legge inn perioder hvor de er opptatt eller ønsker fri.

Denne funksjonen ble lagt til vaktplan, fordi her får brukerne en oversikt og kan administrere vaktene sine. Funksjonen er lagt i et skjema som er skjult. Dette er fordi det er ikke et skjema som skal brukes hver gang en bruker besøker siden. Vi ønsket likevel at den skulle være godt tilgjengelig, derfor kan skjemaet synliggjøres ved hjelp av en knapp øverst på siden.



Figur 40, Skjema for å sette status for en tidsperiode

For å realisere dette systemkravet måtte det gjøres nye endringer til databasen. En tabell som ble opprettet under Sprint 2 måtte endres. Grunnen var at koblingstabellen *ansatt_vakt_status* kobler idvakt, idansatt og idstatus. Men nå skulle status kobles til idansatt og dato siden vi ønsker å sette status for en periode uavhengig om en ansatt har vakt i perioden eller ikke. Under vises tabellene før og etter endringene.

	idvakt	idansatt	idstatus
	54	41	1
	55	41	2
▶	56	41	3

Figur 41, Databasetabell ansatt_vakt_status

idansatt_status	id_ansatt	id_vakt	dato	id_status
37	41	NULL	2019-04-12	2
38	41	NULL	2019-04-13	2
39	41	NULL	2019-04-14	2
40	41	37	NULL	2

Figur 42, Databasetabell ansatt_status

Her har tabellen fått en egen primærnøkkel (identifikator) fordi både id_vakt og dato kan være null, altså ikke ha verdi, og kan derfor ikke være med i en kombinert primærnøkkel. Kombinasjonen av id_ansatt og id_status er ikke identifiserende og fungerer derfor ikke som primærnøkkel. Med disse endringene kan tabellen både brukes til å sette status på en enkelt vakt, derav kolonnen id_vakt, eller for en dato/periode.

3.4.4. Avslutning av implementeringsfasen

Etter Sprint 3 ser vi at de aller fleste systemkravene er fullført. Vi er godt fornøyd med resultatet, og følte at struktureringen av arbeidet i tre sprinter har gjort utviklingen av applikasjonen effektiv.

Etter vi var ferdig med utviklingen av applikasjonen har vi brukt tid på å teste applikasjonen. Testingen ble utført av gruppemedlemmene. Den ferdige applikasjonen ble vist frem til oppdragsgiver på et møte (se møtereferat 8.7.4). Han har ikke tatt applikasjonen i bruk, men sier at den ser bra ut, og han tror den kan forenkle arbeidsdagen hans.

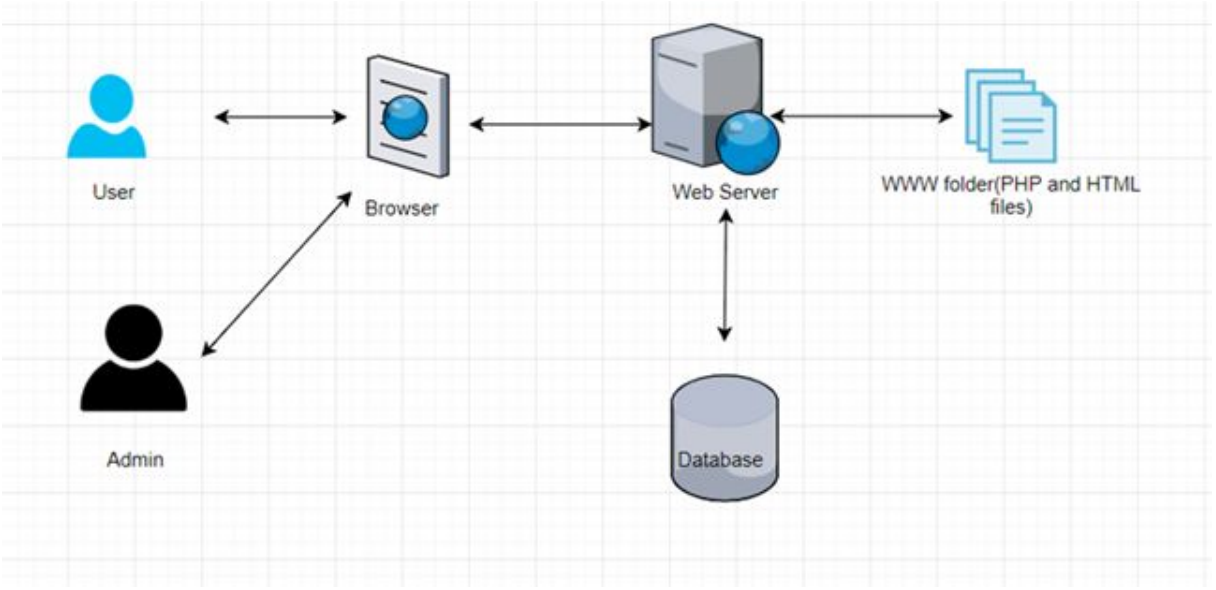
Vi har i tillegg laget en brukermanual (se vedlegg 8.9), som er tenkt å være et oppslagsverk for oppdragsgiver og andre brukere av applikasjonen.

4. Systemdokumentasjon

Denne den delen av rapporten dokumenterer vi systemarkitektur, database, sikkerhet og til slutt utforming av applikasjonen.

4.1. Systemarkitektur

Systemarkitekturen viser strukturen og oppførselen av systemer. En slik arkitektur viser hvordan komponentene er koblet til hverandre og hvordan dataene blir sendt mellom (Spacy, 2018).



Figur 43, Systemarkitektur

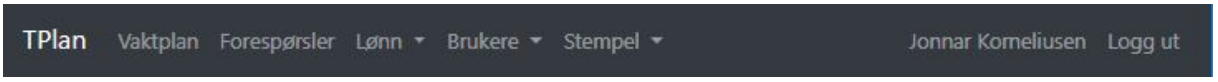
Modellen ovenfor viser når User (ansatt) og Admin (administratoren) åpner applikasjonen i en nettleser, her kalt Browser. De skriver nettadressen til vår TPlan.no i deres adressefelt og det blir sendt en forespørsel til webserveren. PHP-filen index.php, i likhet med andre HTML- og PHP-filer som utgjøre applikasjonen vår, ligger på webserveren og blir sendt til nettleseren som leser og tolker koden, og viser den til brukeren. Fra index.php må alle brukere logge seg inn for å få tilgang til applikasjonen bak. Webserveren sørger også for kommunikasjon mellom PHP-koden og databasen.

4.2. Utforming

Her beskriver vi de ulike valgene vi tok for å utforme webapplikasjonen slik at den ble brukervennlig for våre brukere.

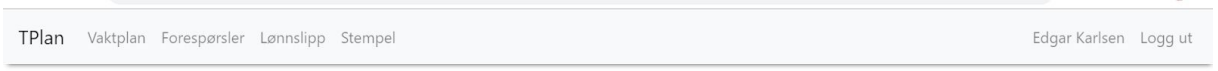
Navigasjonsbarer

Vi har brukt ulike farger på navigasjonsbarene til bruker og administrator. Dette er for å gi et tydelig skille mellom de to delene av applikasjonen.



Figur 44 Navigasjonsbar administrator

Figur 45 Navigasjonsbar administrator på liten skjerm

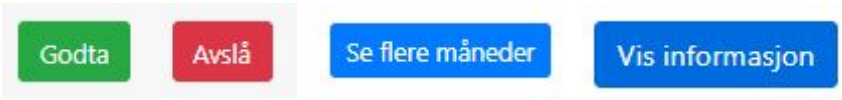


Figur 46 Navigasjonsbar bruker på stor skjerm

Figur 47 Navigasjonsbar bruker på liten skjerm

Bruk av farger

Vi har bevisst benyttet oss av ulike farger for å signalisere til brukeren ulike budskap. I knapper rød og grønn for å tydeliggjøre hva knappen gjør. Grønn godkjenner mens rød avslår, deaktiverer eller sletter. Blå knapp henter fram mer informasjon.



Figur 48, Knapper

For å skille mellom vakter som er ledige, og de som er dekket, har vi valgt å bruke rosa farge på ledige vakter og grønn på egne og andres vakter.

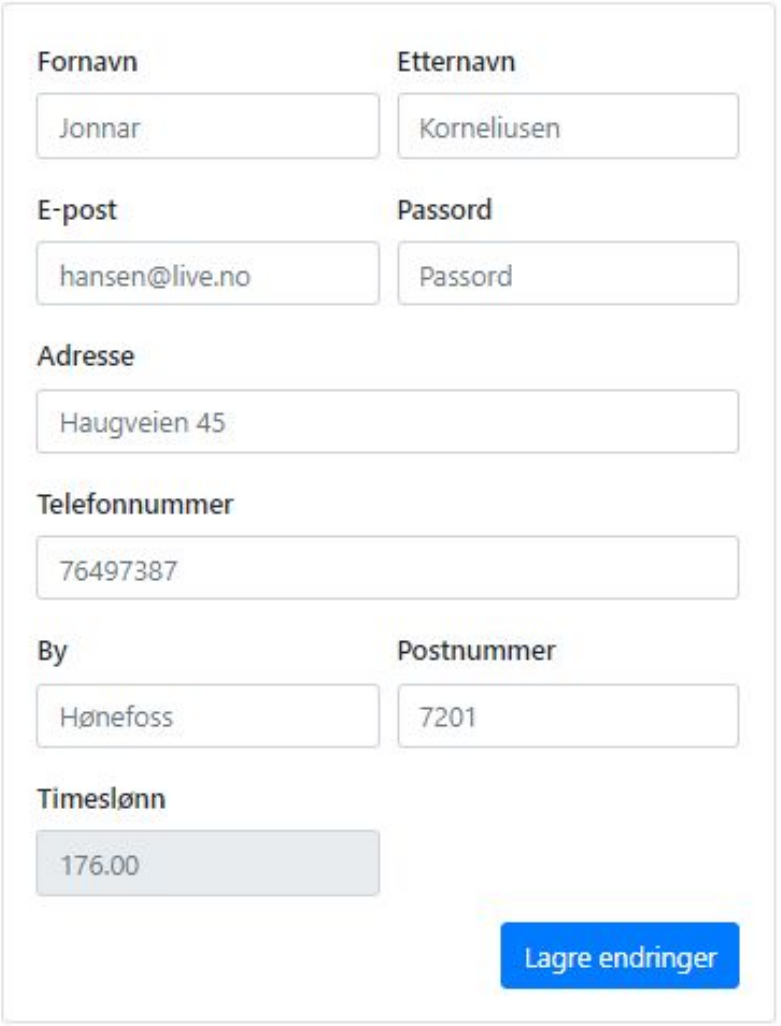
Figur 49 Bruk av fargekode på vakter

For å varsle administrator om ledige vakter innenfor de neste fem dagene valgte vi å bruke en gul stripe som symboliserer en mild advarsler.

Figur 50 Gul advarsel

Skjemaer

Alle skjemaene, fra innlogging til min profil, i applikasjonen består av samme designelementer. Dette gjør at applikasjonen får en designmessig rød tråd slik at brukerne kjenner seg igjen i de ulike skjemaene.



Figur 51 Eksempel på skjema

4.3. Database

Databasen vi bruker har forskjellige tabeller som vist ovenfor, noen av disse tabellene har relasjoner til hverandre. Dataene er relevante for applikasjonen, men ikke alle tabellene er like selvforklarende. Nedenfor vil vi skrive kort om de ulike tabellene, hvilken type data de skal inneholde og hva hensikten med de er.

Tabellen *ansatt* inneholder all informasjon om ansatte, her bruker vi "id_ansatt" som hovednøkkel, dette betyr at dette feltet er unikt og identifiserende og i vårt system skal denne nøkkelen generes automatisk. Dette er for å hindre dobbeltlagring av data og samtidig kunne skille mellom ansatte. Denne tabellen har relasjoner til flere andre tabeller i databasen, hvor "id_ansatt" blir brukt som en fremmednøkkel for å kunne identifisere hvilken ansatt det er snakk om. For eksempel så bruker vi "id_ansatt" i vaktplan tabellen, dette er for å kunne vite hvilken ansatt som er satt opp på vakten.

Dataen som blir lagret i tabellen *bruker* blir brukt til innlogging og for å kunne differensiere om brukeren har administratorrettigheter eller ikke. Her bruker vi "id_ansatt" for å knytte dataen til riktig ansatt, og "admin" feltet for å finne ut om brukeren har administratorrettigheter. Dersom feltet inneholder verdien 0 betyr dette at det er en vanlig bruker, dersom den inneholder verdien 1 betyr det at brukeren er administrator.

Meldinger til oppslagstavla blir lagret i tabellen *melding*. Her lagrer vi identifikasjon på meldingen, melding, hvilken ansatt den er sendt av og når den ble sendt.

Tabellen *vaktplan* består av planlagte vakter som administrator har opprettet. Den inneholder en "vakt_id" som skiller vaktene fra hverandre. Tabellen inneholder også dato og klokkeslett på når vakten starter og slutter, samt en kort beskrivelse av vakten. Merk at dersom "id_ansatt" feltet er tomt, vil det bety at vakten er ledig.

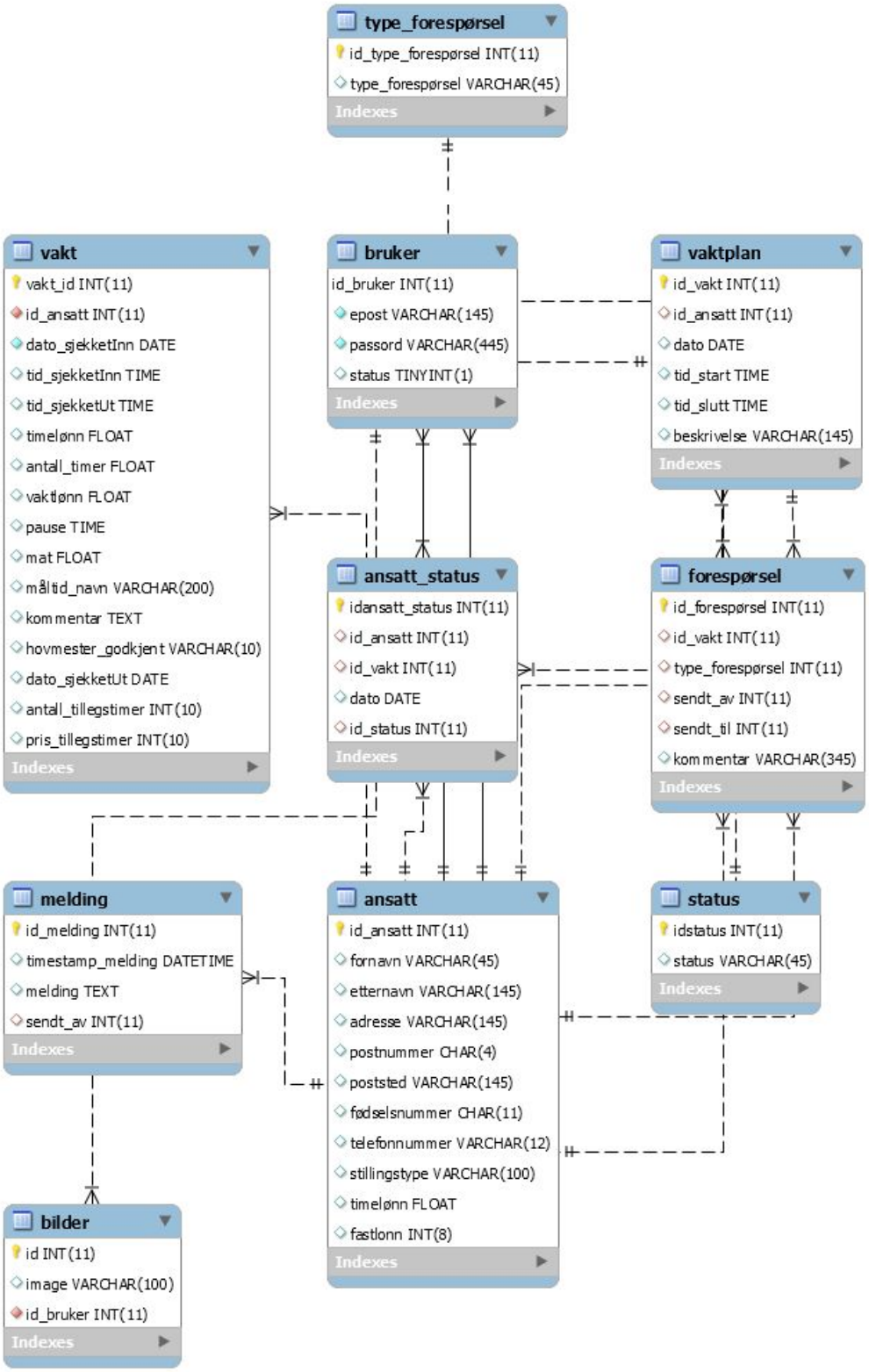
Timeregistrering og loggføring av ansatte sine vakter skjer i tabellen *vakt*. Hver gang en ansatt sjekker inn registreres det i denne tabellen. Ved utsjekk registrerer de ansatte hvor lang pause de har hatt, og om de har spist og verdien på måltidet. De kan også legge til kommentar. Denne dataen bruker vi for å regne ut hvor mange timer en ansatt har jobbet og

hvor mye en ansatt har tjent i løpet av hele vakten. Denne tabellen er essensiell til å kunne vise ansatte en oversikt av månedens lønn, og for å at daglig leder skal kunne beregne lønnen til de ansatte på slutten av måneden.

I tabellen *forespørsel* lagres alle forespørsler. En forespørsel kan være ønske om fri, eller å selge en vakt. Som vist på modellen så har denne tabellen relasjoner til tre tabeller *ansatt*, *vaktplan* og type*_forespørsel*. Dette er fordi den trenger informasjon om hvilken ansatt og vakt forespørselen gjelder, og hvilken type forespørsel det er. Det kan da være "Selge vakt" eller "Ønsker fri". Ansatte kan også legge til en kommentar om hvorfor de sender forespørselen, dersom en ansatt vil bytte vakt kan den sende forespørselen direkte til kollegaen det gjelder. Typen forespørsel er lagret databasen for at administrator skal kunne legge til eller slette forskjellige typer, og vi kan sortere og vise fram de ulike typene på ulike måter i applikasjonen.

Vi har opprettet testdata som vi kommer til å ta i bruk under utviklingen. Se vedlegg 8.7 Database for mer informasjon og for å se hvordan dataene ser ut i databasen.

Figur 52 nedenfor viser en modell av databasen slik den ser ut per mai 2019.



Figur 52, Oppdatert databasemodell

4.4. Sikkerhet

Vi har utviklet ulike sikkerhetstiltak for å sikre applikasjonen konfidensiell og sensitive data. Alle systemer bør inkludere autentisering, som for eksempel at folk må logge seg inn før de kan navigere rundt på nettsiden eller applikasjonen. Videre bør det deles ulike rettigheter i forhold til deres posisjon i organisasjonen, for eksempel at sjefen og de ansatte bør ikke ha samme rettigheter i systemet. De bør skilles, slik at det skaper mer sikkerhet (Viega & McGraw, 2002, s.20 og s. 22) .

Kryptering av passord

Å kryptere passord er en sikkerhetstiltak som vi benyttet. Vi valgte å «hashe» og «salte» passord slik at passordet ikke skal kunne leses i klartekst. Bilde under viser hvordan vi krypterte passordet ved å bruke "hashing" og "salting" metoden.

```php
if ($send_data == true){
    echo "send data";


    $salt="pastacafe2000";
    $saltet_pass = $passord.$salt;
    $saltet_pass = sha1($saltetpass);
```

Figur 54, Kryptering av passord

Når passordet blir «hash-et» blir det ved hjelp av en algoritme som gjør at passordet ikke blir lagret i et klartekst. Under er et bilde av hvordan en kryptert passord ser ut i databasen vår.

id_bruker	epost	passord	status
41	karlsen@live.no	da39a3ee5e6b4b0d3255bfef95601890afd80709	0
42	hansen@live.no	da39a3ee5e6b4b0d3255bfef95601890afd80709	1
60	balerud@live.no	da39a3ee5e6b4b0d3255bfef95601890afd80709	1

Figur 55, Resultat av kryptering

Passordlengde

Vi har både maksimum- og minimums krav på passord inputfeltet vårt i endring og registrering av nye brukere. Passordet må ha minst åtte tegn eller flere tegn for at passordet blir godkjent.

Tiltak mot SQL-planting

Vi har benyttet oss av de innebygde funksjonene som PHP har mot. Ved å kjøre SQØ-spørringene gjennom såkalte *prepared statements* håper vi å sikre applikasjoner mot hackere som prøver å tilegne seg passord, eller på ulike måter endre eller slette data i databasen. Figur 56 viser koden for *prepared statements* ved innlogging.

```
$sql = "SELECT * from tplanno01.bruker";
$sql .= " WHERE epost = ?";
$sql .= " AND passord = ?";

$statement = $db-prepare($sql);

$statement->bind_param("ss", $_POST['mail'], $saltet_pass);
$statement->execute();
$statement->bind_result($id);
```

Figur 56 Kode med prepared statement

Personvern

Vårt system inneholder sensitive informasjon, dermed er det viktig å tenke på hvordan disse blir behandlet. Sensitive informasjon er blant annet navn, fødselsnummer og e-post. Digitalt personvern blir håndhevet av Datatilsynet og norske bedrifter er pålagt å følge EU-direktiver GDPR (General Data Protection Regulation) som omhandler behandling av personlig data. Behandling av sensitive data bør derfor være samtykket på forhånd mellom den ansatte og bedriften (Judin, 2019).

Sikkerhetsplan

Vi er bevisst på at applikasjonen vår ikke er sikret tilstrekkelig til at Pasta Cafe's sensitive data er trygge. Vi mener selv at vi ikke har nok erfaring med sikkerhet og dermed anbefaler vi oppdragsgiver til å kontakte et firma som har mer erfaring med å sikre webapplikasjoner. Vi har derimot utarbeidet en liste med anbefalinger for hva som må gjøres videre for å sikre nettsiden.

1. Sikkerhetskopiering

   Domeneshop, som hoster applikasjonen, tar dessverre ikke ansvar for sikkerhetskopiering av filer. Dermed faller ansvaret på Hussam som må benytte seg av en løsning for å ikke miste viktige informasjon og databasen sin. Hvis man opplever et nettverksangrep eller feil med programvaren, og man mister viktige data, er viktig å ha en kopi. Med sikkerhetskopiering mister man ikke viktige data og dataene er tilgjengelige. Det er ikke sikkert at man får en komplett data tilbake, men det meste får man tilbake. Fordelen med sikkerhetskopiering er at det er mer effektivt og mer sikkerhetsalternativ for å ikke miste data.

   Det finnes løsninger for sikkerhetskopiering som gjør jobben for deg. De tar backup, oppdaterer, og tilbyr antivirus. For eksempel mest kjent sikkerhetskopiering alternativ er Microsoft Azure, SAP eller Cloud sine nettsky alternativer.

2. Brannmur

   En brannmur er et program som beskytter datamaskinen mot nettverksangrep. Brannmuren identifiserer og blokkere skadelige trafikk gjennom nettverket, men den kan ikke stoppe alle typer trusler. De vanligste operativsystemer har allerede brannmur installert. På Windows PC-er er for eksempel *Windows-brannmur* som beskytter forskjellige trusler. Det finnes ulike personlige brannmur kan lastes ned både gratis eller kjøpes.

3. Kontinuerlig overvåking

   Når datamaskinen er koblet til Internett, er det en risiko for at en hacker kan utnytte svakheter i applikasjonen. Kriminelle (hackere/cracker) kan legge igjen en type virus som ormer inn i nettverket og stjeler og/eller endrer viktige data uten ditt samtykke. Nett-skannere jobber og analysere i bakgrunnen for å overvåke applikasjonen og se etter mulige trusler. Noen automatiserte skannere kan til og med fjerne skadelige programvare. Derfor er det viktig med kontinuerlig overvåking av webapplikasjonen for å redusere trusler (Collins, 2014). Vi er usikker på om nettverksskanner er nødvendig for Pasta Cafe, siden de er ikke et stort selskap med mye data og nettverkstrafikk, men det kan være nyttig å undersøke.

   Her er e liste over våre anbefalinger som en helhet for å sikre webapplikasjonen:

   - Malware skanning
   - Fjerne mulige skadelige programvarer
   - Spam overvåking
   - Sikkerhetsovervåking
   - Brannmur
   - Trene ansatte om nettverkssikkerhet

Vi anbefaler også at ansatte blir opplært i hva som kan skje dersom de møter på en trussel både fysisk og digitalt. For eksempel informere de om at de bør ikke trykke på en lenke som de ikke er helt sikker på kom fra en sikker kilde osv. Opplæring av de ansatte kan være med å hjelpe og redusere skader og uønskede hendelser.

Siden TPlan er laget til Pasta Cafe og håndterer de ansattes persondata, er det viktig å sjekke om applikasjonen følger direktiver personvern, for eksempel GDPR. Hvis ikke direktivene følges kan Pasta Café bøtelegges. Derfor er det viktig å sikre om at de har fulgt lover og regler for persondata.

5. Diskusjon

Her diskuterer vi de valgene som vi tok underveis i prosjektet i fellesskap. Vi har delt denne delen i ulike avsnitt, da vi mener dette skaper mer forståelse.

Valg av språk og verktøy

Å velge et utviklingsspråk som vi ikke var kjent med tror vi hadde tatt for lang tid i forhold til å lage en applikasjon som vi er fornøyd med. I tillegg levere et arbeid som kan gi et godt resultat. Vi benyttet PHP fordi, det fungerte godt for å koble sammen databasen og applikasjonen. Hvis vi hadde valgt å bruke et ukjent programmeringsspråk, som f.eks. JavaScript, hadde vi brukt mye tid på å lære oss det, og med PHP dikterte ikke språket hva vi fikk til. Vi kunne kode de funksjonene vi ville, slik som vi ønsket det.

Hele gruppen hadde benyttet seg av Visual Studio Code (VSC) som utviklingsverktøy. Den fungerte veldig godt som integrasjon mot GitLab, og koden vår var alltid oppdatert og tilgjengelig for alle gruppemedlemmene. På den andre siden gjorde VSC det vanskelig for oss å feilsøke koden da den baserte seg på utvidelser, ofte kodet av brukere, for å gi støtte for spesifikke utviklingsspråk. Flere av disse utvidelsene fungerte ikke slik vi ønsker, og slik vi var vant til fra andre utviklingsverktøy vi har brukt.

Vi er fornøyd med Domeneshop som hosting-verktøy. Vi hadde i utgangspunktet ikke behov for hosting da vi kunne hatt applikasjonen på en skole-server, men da hadde vi ikke hatt tilgang til applikasjonen andre steder enn på campus. En stor ulempe med Domeneshop var at de ikke ga oss mulighet til å sette automatiske prosedyrer på databasen, som f.eks. sette brukere som "logget ut" ved midnatt. Dette løste vi med alternative funksjoner skrevet i PHP.

Arbeidsprosessen

Gruppearbeidet har fungert bra under hele prosjektet. Alle har møtt opp og vist engasjement i å levere et godt arbeidet. På grunn av ulike livssituasjoner har det vært viktig å være fleksible og kommunisere godt. I utgangspunktet møtte vi fast to dager i uken, men av og til ble det endringer, slik at det passet best for enhver gruppemedlemmer. På grunn av god planlegging

og fordeling av arbeid har det ikke vært vanskelig å kombinere arbeidet med bacheloroppgaven, med andre fag og andre aktiviteter. Motivasjonen til gruppemedlemmene har gått i bølgedaler. Vi synes arbeidet går bedre og det er lettere å motivere seg når vi møtes og jobber sammen. Vi er glad for at arbeidet med bacheloroppgaven har vært gruppearbeid og vi tror ikke vi hadde klart å produsere en applikasjon og en rapport av like høy standard på egenhånd.

Bruk av Scrum

Vi har ikke fulgt metoden Scrum til punkt og prikke fordi det ikke passet måten vi jobbet på. For eksempel har vi ikke hatt daglige Stand-up, siden vi ikke møttes hver dag, men vi har vært flink å dele oppgaver og informerte hverandre dersom et problem oppstod. Sprintene hjalp oss med å få innsikt i hverandres arbeidsprosess. Hver gang en ny Sprint begynte holdt vi planleggingsmøte. Her så vi på tidsplanen hva som var gjort og hva som gjenstod. I tillegg om teammedlemmene støttet på problem(er). De systemkravene som var høyest prioritert var identifisert i kravlisten, og de ble selvsagt utviklet først. De første Sprintene fordelte vi litt for lite oppgaver, men etter hvert når vi fikk bedre forståelse av rapporten og oppgaven, ble Sprintene mer effektivt, og vi ble flinkere med å fordele oppgavene i henhold til sprintens varighet. God planlegging av sprintene gjorde at vi ikke var avhengig av å møtes hver dag, da hvert av oss hadde egne oppgaver å jobbe med. Scrum masteren hadde kontakt med veileder og passet på at alle teammedlemmene forstod og gjorde sin del av oppgaven som var delt i fellesskap. Hvis vi ikke møttes fysisk, brukte vi sosiale medier og Google Docs for å kommunisere.

Scrum ga oss muligheten til å jobbe fritt, samt. man har rammer og reglene som gjør at man prioriterer noen funksjoner, og har en god oversikt over de andre funksjonene, slik at man ikke glemmer de. Utnyttelsen av denne smidige metoden gjorde at vi startet med dette prosjektet tidlig, selv om det var litt uklar hvor prosjektet skulle føre oss til slutt. Som nevnt tidligere benyttet vi oss av Google Docs og Facebook messenger som samarbeidsverktøy, fordi de er enkle i bruk. Det samme synes vi gjelder for metoden vår - en liste over

systemkrav og en tabell for hver sprint i implementeringsfasen over hva som skal gjøres. Det er enkelt og har lav terskel for å bli brukt.

Applikasjonen

Vi er fornøyd med applikasjonen vi har utviklet. Systemkravene under alle MÅ-kravene ble utviklet. Videre ble også de fleste BØR-kravene utviklet. Slik applikasjonen er nå føler vi at det er mer vi kunne gjort og den har forbedringspotensialet. Med den tidsrammen vi hadde var det vanskelig å få gjennomført alle kravene og alle ideene vi hadde.

Rapporten

I starten av prosjektet bestemte vi oss for å fokusere på dokumentasjon og rapporten. Dette førte til at mye av arbeidet med rapporten var gjort, og når vi var ferdig med utvikling av systemkravene. Da gjenstod dokumentasjon av systemet, brukerveiledning, diskusjon og oppsummering. Dette gjorde at vi kunne fokusere på koding og utvikling i implementeringsfasen.

6. Oppsummering

Når vi ser tilbake på prosjektet vi er fornøyd med prosessen, samarbeidet og sluttproduktet. Prosjektgjennomføringen gikk etter planen og vi føler at vi har hatt god fremgang gjennom hele prosjektet. Utviklingsmetoden som vi benyttet oss av har bidratt til å levere et godt resultat. Den gjorde at vi kunne jobbe fleksibelt, fordele oppgaver effektivt og vi klarte å oppfylle viktige systemkrav tidlig i implementeringsfasen.

Samarbeidet i gruppen har gått bra og vi har vist tillit innad gruppen når det kommer til oppmøte og utføring av oppgaver. Fordelen med dette er at vi har lært å forholde oss til hverandre og stole på at alle brikker faller på plass på slutten av hver sprint. Gruppemedlemmer har vært flinke til å gi og ta imot konstruktive tilbakemeldinger, alle meninger ble hørt og tatt til vurdering under diskusjoner.

Daglig leder av Pasta Cafè ville gå over fra papirformat til et system der han kan fullføre hans kontor-arbeidsoppgaver på en effektivt måte. Vi føler at vi har klart å utviklet et slikt system, som han kan benytte seg av. Systemet hjelper han blant annet med å registrere/endre vakter, se lønnsslipp til hver enkel ansatte og få en oversikt over vaktplanen. I tillegg får ansatte en oversikt når de er satt opp til å jobbe, og har muligheten til å melde seg til ledige vakter, endre eller selge vakten deres. Dette skaper en smidig kommunikasjon mellom daglig leder og ansatte. Med dette systemet kan oppdragsgiver spare tid og ressurser for å fullføre hans daglige kontor-arbeidsoppgaver.

Dette har vært en utfordrende, morsom og lærerik prosess. Vi har hatt et godt samarbeid, og vært en gruppe som støtter og viser hensyn til hverandre.

7. Referanser

Litteraturliste

Felke-Morris, T. (2012). *Basics of Web Design: HTML5 & CSS3*. Boston: Addison-Wesley.

Horgen, S. A. (2014). *Webprogrammering i PHP* (4. utg.) Oslo: Stiftelsen TISIP og Gyldendal Akademiske.

Holmstedt, Viggo. (2015). *Objektorientert systemutvikling og UML* (2.utg.). Bergen: Fagbokforlaget.

Kristoffersen, B. (2016). *Databasesystemer* (4. utg.). Oslo: Universitetsforlaget.

Roth, R. M., Dennis, A., & Wixon, B. H. (2013). *Systems Analysis and Design: International Student Version*. Singapore: John Wiley & Sons.

Skagestein, Gerhard. (2002). *Systemutvikling - fra kjernen og ut, fra skallet og inn.* Kristiansand: Høyskoleforlaget.

Sandnes, Frode Eika. (2011). *Universell utforming av IKT – systemer. Brukergrensesnitt for alle.* Oslo: Universitetsforlaget.

Nettsider

Admin. (2018). *What is website security?*. Hentet 15.05.2019, fra: https://cwatch.comodo.com/blog/website-security/what-is-website-security/

Adolfsen, Janniche. (2015). *Responsivt design – hva og hvorfor?*. Hentet 24.01.2019, fra: https://www.idium.no/blogg/2015/responsivt-design-hva-og-hvorfor/

Aven, Terje (2016). *Risikoanalyse*. Hentet 02.12.2018, fra: https://snl.no/risikoanalyse

Bootstrap. (u.å.) *Licence FAQs.* Hentet 16.04.2019, fra: https://getbootstrap.com/docs/4.0/about/license/

Collins, Alex. (2014). *The Important of Vulnerability Scans.* Hentet 15.05.2019, fra: https://www.allcovered.com/the-learning-center/the-importance-of-vulnerability-scans-729

Crestodina, Andy. (u.å.). *Web Design Standards: 10 Best Practices on the Top 50 Websites.* Hentet 26.02.2019, fra: https://www.orbitmedia.com/blog/web-design-standards/

De nasjonale forskningsetiske komiteene. (2010). *Kvalitative og kvantitative forskningsmetoder – likheter og forskjeller*. Hentet 02.04 2019, fra: https://www.etikkom.no/forskningsetiske-retningslinjer/medisin-og-helse/kvalitativ-forskning/1-kvalitative-og-kvantitative-forskningsmetoder--likheter-og-forskjeller/

Direktoratet for forvaltning og ikt. (2019, 10.januar). *Kva er universell utforming?*. Hentet 24.01.2019, fra: https://uu.difi.no/kva-er-universell-utforming

Difi. (2018). *Kva seier forskrifta?* Hentet 24.02.2019, fra: https://uu.difi.no/krav-og-regelverk/kva-seier-forskrifta

Difi, Universell utforming. (2015). *Navigasjonsmetoder.* Hentet 24.01.2019, fra: https://uu.difi.no/krav-og-regelverk/losningsforslag-web/navigasjonsmetoder

Difi, Universell utforming. (2015). *Navigasjonsmetoder.* Hentet 24.01.2019, fra: https://uu.difi.no/krav-og-regelverk/losningsforslag-web/bruk-av-farger

Difi. (2016). *WCAG 2.0.* Hentet 29. 01.2019 fra: https://wcag.difi.no/wcag-20.html

Federl, M. (2019). Møteinnkalling og møtereferat - NDLA. Hentet 03.02.2019, fra https://ndla.no/subjects/subject:19/topic:1:195989/topic:1:195922/resource:1:19085

Experience. (u.å.) *What is Wireframing.* Hentet 20.02.2019, fra: https://www.experienceux.co.uk/faqs/what-is-wireframing/

Gangåssæter, Knut. (2016). *Har du god navigasjon på ditt nettsted?*. Hentet 24.01.2019, fra: http://doghouse.no/2016/02/18/1514/

GitLab. (2019). *What is GitLab.* Hentet 01.02.2019, fra: https://about.gitlab.com/what-is-gitlab/

Glasspaper. *Hva er egentlig Scrum?*. Hentet 08.02.2019, fra: https://www.glasspaper.no/artikkel/hva-er-egentlig-scrum/

Gramstad, T. (2018). *Åpen kildekode - Store norske leksikon*. Hentet 02.04.2019, fra https://snl.no/åpen_kildekode

Gundersen, Christer. (2017). *Layout.* Hentet 24.01.19, fra: https://ndla.no/subjects/subject:1/topic:1:172416/topic:1:186407/resource:1:163007

Helle, Maren. (2013). *En kort introduksjon til Scrum.* Hentet 08.02.2019, fra: https://www.visma.no/blogg/en-kort-introduksjon-til-scrum/

Joomla!. (u.å.) *Hva er MySQL.* Hentet 01.02.2019, fra: https://www.joomlainorge.no/foreningen/om-joomla?start=4

Judin, Tobias. (2019). *Ikke mulig å omgå personvernreglene*. Hentet 16.04.2019, fra: https://www.personvernbloggen.no/2019/04/15/ikke-mulig-a-omga-personvernreglene/

Khillar, Sagar. (2018). *Difference between JavaScript and PHP.* Hentet 20.02.2019, fra: http://www.differencebetween.net/technology/difference-between-javascript-and-php/

Kolowich, Lindsay. (2018). *Web Design 101: How HTML, CSS, and JavaScript work.* Hentet 20.02. 2019, fra: https://blog.hubspot.com/marketing/web-design-html-css-javascript

Microsoft. (2019). *Why did we build Visual Studio Code?*. Hentet 01.02.2019, fra: https://code.visualstudio.com/docs/editor/whyvscode

Microsoft Azure. (2019). *Visual Studio - kode.* Hentet 20.02.2019, fra: https://azure.microsoft.com/nb-no/products/visual-studio-code/

Nadeau, Michael. (2018). *General Data Protection Regulation (GDPR): What you need to know to stay compliant*. Hentet 15.05.2019, fra: https://www.csoonline.com/article/3202771/general-data-protection-regulation-gdpr-requirements-deadlines-and-facts.html

Otto, M., & Thornton, J. (n.d.). *About Bootstrap*. Hentet 02.04.2019, fra: https://getbootstrap.com/docs/4.3/about/overview/

Planday. (u.å.). *Planday.* Hentet 20.02.2019, fra: https://www.planday.com

Proffhosting. (u.å.). *Hva er webhotell.* Hentet 23.04.2019, fra: https://proffhosting.no/knowledgebase/11/Hva-er-et-Webhotell.html

Sander, Kjetil. (2018). *Prosjektmodell.* Hentet 22.01.2019, fra: https://estudie.no/prosjektmodell/

Spacy, John. (2018). *What is System Architecture?*. Hentet 17.04.2019, fra: https://simplicable.com/new/system-architecture-definition

w3counter. (2018). *Web Browser Market Share*. Hentet 31.12.2018, fra: https://www.w3counter.com/globalstats.php .

W3schools. (u.å.). *CSS Introduction.* Hentet 01.02.2019, fra: https://www.w3schools.com/css/css_intro.asp

Warner, Adam. (2018). *5 Simple Website Security Best Practices.* Hentet 15.05.2019, fra: https://www.sitelock.com/blog/2018/04/5-simple-website-security-best-practices/

Bilde/Figur

Bootstrap. (u.å.). *Bootstrap*. Hentet 02.04.2019, fra: https://getbootstrap.com

Butler, Nick. (2018). *What is Scrum? The Product Owner primer.* Hentet 08.02.2019, fra: https://www.boost.co.nz/blog/2018/03/what-is-scrum-product-owner-primer

Cosulotuion. (2018). *gitlab.* Hentet 02.04.2019, fra: http://www.cswp.at/karriere-portal/web-dev/attachment/gitlab/

Domeneshop. (u.å.). *Domeneshop.* Hentet 02.04.2019, fra: https://www.domeneshop.no

Icon-icons. (u.å.). *Icône CSS3 Gratui*t.Hentet 02.04.2019, fra: https://icon-icons.com/fr/icone/CSS3/102605

Trakhtenberg, Arnold. (2018). *Launched: Version 2 of the LauchDarkly Visual Studio Code Extension.* Hentet 02.04.2019, fra: https://launchdarkly.com/blog/launched-version-2-of-the-launchdarkly-visual-studio-code-extension/

Visual Paradigm. (u.å.). *Accessibility of the web-based features.* Hentet 02.04.2019, fra: https://www.visual-paradigm.com/features/vp-online/desktop-accessibility/

Wikipedia. (2001). *PHP.* Hentet 02.04.2019, fra: https://no.wikipedia.org/wiki/PHP

Wikipedia. (2011). *HTML5*. Hentet 02.04.2019, fra: https://en.wikipedia.org/wiki/HTML5#/media/File:HTML5_logo_and_wordmark.svg

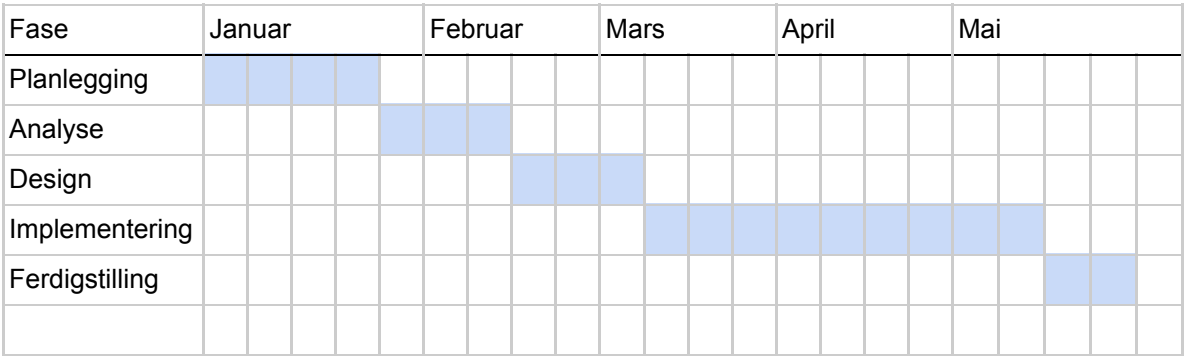
Wikipedia. (2018). *MySQL*. Hentet 02.04.2019, fra: https://en.wikipedia.org/wiki/MySQL#/media/File:MySQL.svg

8. Vedlegg

8.1. Planleggingsfasen

GANTT

Visuell fremvisning av prosjektplanen ved hjelp av et Gantt-skjema:



Figur 57, Gantt-diagram

8.2. Risikoanalyse

- Grønn (Lav) – Enkelt å håndtere og fører ingen fare for prosjektets gjennomføring.
- Oransje (Middels) – Kan skade prosjektet og ødelegge effektiviteten, men kan kanskje løses.
- Rød (Høy) -Vil føre til større risiko og bidra til ødeleggelse av prosjektets mål.

Problemer	Risikonivå	Konsekvens	Tiltak
Ustabilitet nettverk kan risikere effektiviteten eller blir utsatt for videre arbeid	Lav	Middels	Uten internett tilgang, vil ikke vår server være aktiv og da får vi ikke sett sluttresultatet av kodebiten vi har jobbet med.
Sykdom/fravær	Lav	Middels	En konsekvens for dette kan føre til at progresjonen i arbeidet reduseres. Et tiltak for dette er å inndele arbeidet likt mellom oss i gruppen. Det er en selvfølge at enhver av oss i gruppen informerer hverandre om fravær eller statusen på arbeidet sitt.
Dårlig arbeidsklima	Lav	Middels	Disse kan løses ved både god kommunikasjon i gruppen og ha regelmessige møter.
Sitter låst i prosessen i rapporten/koding	Lav	Middels	Her må samtlige i gruppen være ansvarlig å melde ifra om bistand fra andre medarbeidere, og aktivt bruke relevante kilder og ressurser for problemløsning.

Urealistisk planlegging av bacheloroppgaven	Lav	Middels	Arbeidsprosessen tar lengre tid å fullføre i forhold til prosjektplanen. Dermed er det sentralt å ha riktig fokus og holde oss til tidsplanen.
Tap av kode/rapport	Middels	Stor	Kan skyldes av menneskelig eller teknologi svikt. Vi tviler at dette vil være et problem. Som et sikkerhetstiltak foretar vi sikkerhetskopiering av dokumentene våre regelmessig.
Oppdragsgiver ombestemmer seg	Lav	Stor	Bryter kontrakten, eller eventuelt endre på kravene. God kommunikasjon med oppdragsgiver er sentralt for å få et stabilt forhold, for å unngå konflikter og misforståelser.
Implementering	Lav	Middels	Oppdragsgiveren blir ikke fornøyd med systemet eller nye endringer som er innført. Et tiltak der vi innkaller oppdragsgiver til et evalueringsmøte for å diskutere endringer og implementeringer.

Figur 58, Risikoanalyse

8.3. Systemkrav

I denne listen under om systemkrav, refererer vi daglig leder som administrator-bruker av systemet.

Funksjonelle krav:

Prosessorientert (en prosess systemet må utføre) og informasjonsorienterte (informasjon systemet skal inneholde)

1. Melde seg opp til, og bytte, vakter.
    1.1. Ansatte skal kunne logge inn med personlig brukernavn og passord.
    1.2. Ansatte skal kunne melde seg ledig eller opptatt til vakter.
    1.3. Ansatte skal kunne legge inn ønske om å bytte vakter, om de begge er ledig den vakten de ønsker å bytte til seg.
    1.4. Ansatte skal kunne legge inn ønske om fri fra en vakt de allerede har registrert på seg.
    1.5. Ansatte skal kunne legge inn tidsperioder hvor de er opptatt eller ønsker fri.
2. Administrere vakter
    2.1. Daglig leder skal kunne opprette eller endre vakter
    2.2. Daglig leder og de ansatte skal kunne se vaktplan.
    2.3. Daglig leder og de ansatte skal kunne se hvilke vakter som ikke er dekket de neste 10 dagene
    2.4. Daglig leder skal bli varslet om vakter som ikke er dekket de neste 5 dagene.
    2.5. Daglig leder skal kunne se hvilke vakter de ansatte ønsker seg fri fra, og eventuelt godkjenne disse.
    2.6. Ansatte skal bli varslet når ønsker om vakter, bytte av vakter og fri er blitt godkjent.
    2.7. Ansatte skal kunne "sjekke inn og ut" av jobb med stemplingsur.
    2.8. Ansatte skal kunne registrere pause.
    2.9. Ansatte skal kunne registrere mat.

2.10. Ansatte skal kunne se vakter, pause, mat og foreløpig estimert lønn som er registrert på dem, og kunne legge inn kommentarer på om opplysningene er riktig eller ikke.

2.11. Administrator skal kunne godkjenne alle som har sjekket inn og endre på dataen dersom det er behov.

2.12. Administrator skal ha egen side der han kan se hvem som er på jobb i nåværende tidspunkt.

3. Lønnsadministrasjon
   3.1. Daglig leder skal, når som helst, kunne se en estimert lønnsutbetalinger.
   3.2. Daglig leder skal ved slutten av hver måned se en oversikt over månedens lønnsutbetaling.
   3.3. Daglig leder skal kunne se detaljerte lønnsoversikt for hver enkelt ansatt (lønnsslipp).
4. Personaladministrasjon
   4.1. Daglig leder skal kunne logge seg inn med brukernavn og passord som gir tilganger som er forbeholdt for administrator
   4.2. Daglig leder skal kunne opprette ny bruker i systemet.
   4.3. Daglig leder skal kunne endre alle opplysninger om en bruker.
5. Rapportering og arkivering
   5.1. Daglig leder skal kunne samle alle opplysninger som revisor trenger i et dokument som enkelt kan sendes videre.

8.4. Bruksmønster

Mal bruksmønster

Navn	
Aktør	
Pre-betingelse	●
Post-betingelse	●
Trigger	
Normal hendelsesflyt	1. 2. 3. 4.
Variasjon	
Relatert informasjon	

Figur 59, Bruksmønster-mal

Navn: Hva heter bruksmønsteret? Bruk aktiv setning ("Ta ut penger")

Aktør: Hvem initierer?

Pre-betingelse: Hva må være oppfylt for at bruksmønsteret skal kunne utføres?

Post-betingelse: Tilstand til system og aktør etter at bruksmønsteret er utført.

Trigger: Situasjon eller hendelse som initierer bruksmønsteret.

Hendelsesflyt: Hva som skjer når alt går bra, når aktøren når målet på enklest mulig måte.

Variasjoner: Unntak fra normal hendelsesflyt.

Relatert info: F.eks. kvalitetskrav.

(Roth, Dennis, Wixom, 2013, s.149)

Vi ser det ikke hensiktsmessig å legge ved alle bruksmønster som er laget, men vi har lagt ved en liste over de bruksmønstre vi har laget og et eksempel.

Liste over bruksmønster:

- Logge på som bruker
- Sette status på vakter
- Legge inn ønske om å bytte vakt
- Legge inn ønske om fri fra vakt
- Logge på som administrator
- Opprette ny bruker
- Endre opplysninger om bruker

Eksempler på bruksmønster:

Navn	Opprette ny bruker
Aktør	Administrator
Pre-betingelse	● Administrator må være innlogget i applikasjonen ● Administrator trenger navn og e-postadressen til den ansatte
Post-betingelse	● Ny bruker blir lagret i databasetabell ● E-post med informasjon blir sendt til bruker ● Applikasjonen krever at bruker skifter passord ved første innlogging
Trigger	Ansatt trenger bruker til applikasjonen
Normal hendelsesforløp	1. Administrator fyller inn skjema for opprettelse av ny bruker. 2. Applikasjonen sjekker at påkrevde felter er fylt ut. 3. Applikasjonen genererer og hasher passord til bruker. 4. Applikasjonen lagrer informasjon om brukeren i databasen 5. Applikasjonen sender e-post med brukernavn, passord, og øvrig informasjon, til den nye brukeren
Variasjon	Administrator fyller ikke inn alle påkrevde felter 1. Applikasjonen gir øyeblikkelig tilbakemelding til administrator om hvilke felter som mangler informasjon.
Relatert informasjon	● I henhold til systemkrav 4.2 ● Passord må lagres kryptert i databasen

Figur 60, Bruksmønster opprette ny bruker

Navn	Sette status på vakter
Aktør	Ansatt
Pre-betingelse	● Ansatt er pålogget applikasjonen
Post-betingelse	● Status for dag / vakt blir lagret i databasen
Trigger	Ansatt ønsker å melde seg til ledig vakt, eller fortelle at den er opptatt til vakt
Normal hendelsesforløp	1. Ansatt velger aktuell dag fra kalenderen 2. Applikasjonen viser vakter for valgt dag. 3. Ansatt setter status a. "Tilgjengelig" eller "utilgjengelig" for spesifikk dag b. "Ønsker" eller "ønsker ikke" for spesifikk vakt
Variasjon	
Relatert informasjon	I henhold til systemkrav 1.2

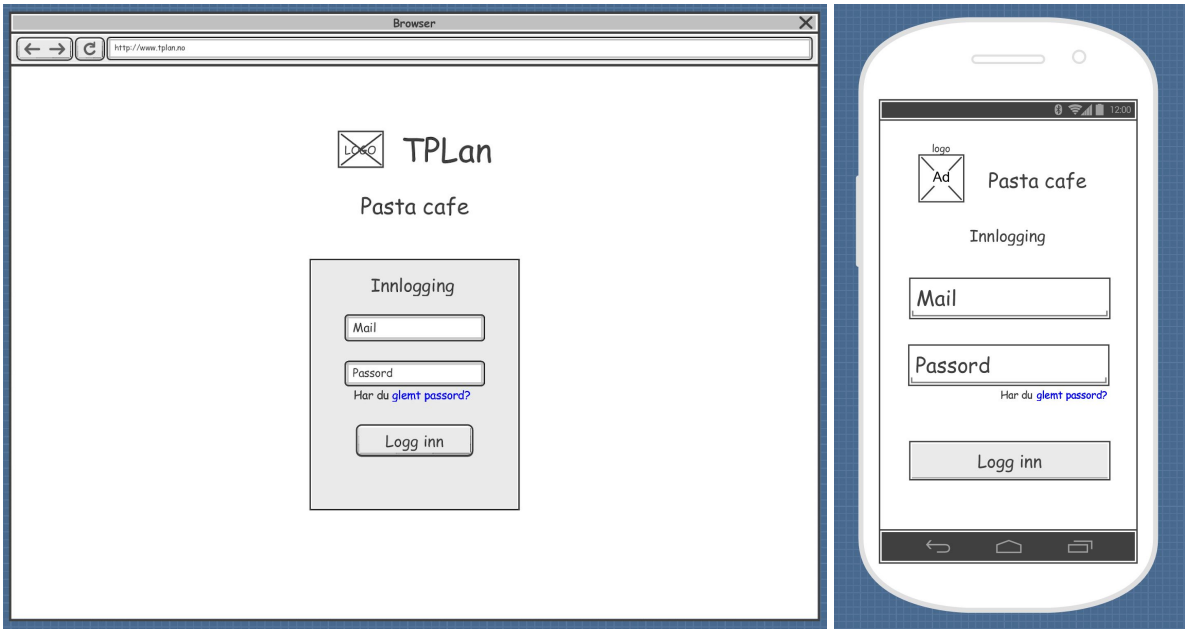
Figur 61, Bruksmønster "sette status på vakter"

Navn	Se lønnsoversikt
Aktør	Administrator
Pre-betingelse	● Administrator må være pålogget applikasjonen ● Dato for siste dag i lønnssyklus må være forbi
Post-betingelse	● Endrede opplysninger blir lagret i databasen
Trigger	Administrator ønsker å se informasjon om lønnsutbetalinger
Normal hendelsesforløp	4. Administrator henter fram skjema for lønnsutbetaling. 5. Applikasjonen fyller feltene med data fra databasen. 6. Administrator kan endre / slette opplysninger. 7. Applikasjonen sjekker om alle påkrevde felter er fylt ut. 8. Applikasjonen lagrer endringene i databasen. 9. Applikasjonen viser lagrede endringer.
Variasjon	Administrator fyller ikke inn alle påkrevde felter 1. Applikasjonen gir øyeblikkelig tilbakemelding til administrator om hvilke felter som mangler informasjon.
Relatert informasjon	I henhold til systemkrav 3.2

Figur 62, Bruksmønster "se lønnsoversikt"

8.5. Wireframes

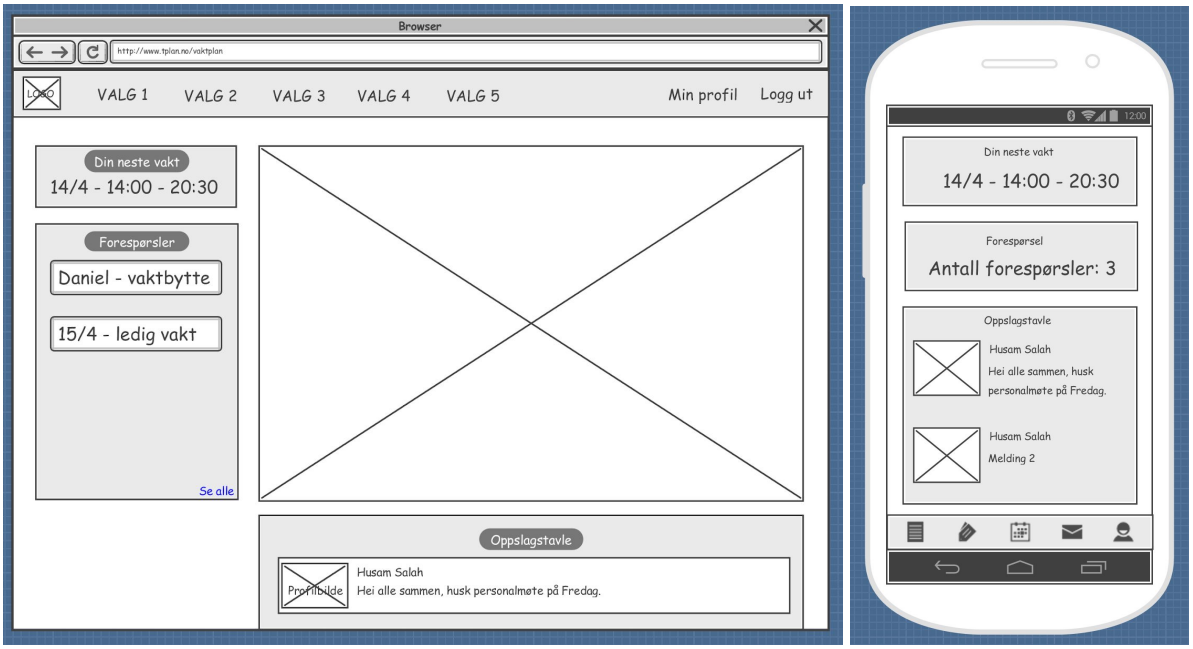
Innlogging for datamaskiner og mobile enheter



Figur 63, Wireframe innlogging

Bildet ovenfor illustrerer hvordan innloggingssiden skal se ut for både mobil og PC enheter. Innloggingssiden er det første en bruker ser og er meget lik på begge enhetene. Bruker kan taste inn sine kredentialer eller bruke linken "Har du glemt passord?" til å tilbakestille passordet ditt. Etter brukeren er logget inn blir den sendt videre til forsiden.

Innlogget forside for datamaskiner og mobile enheter

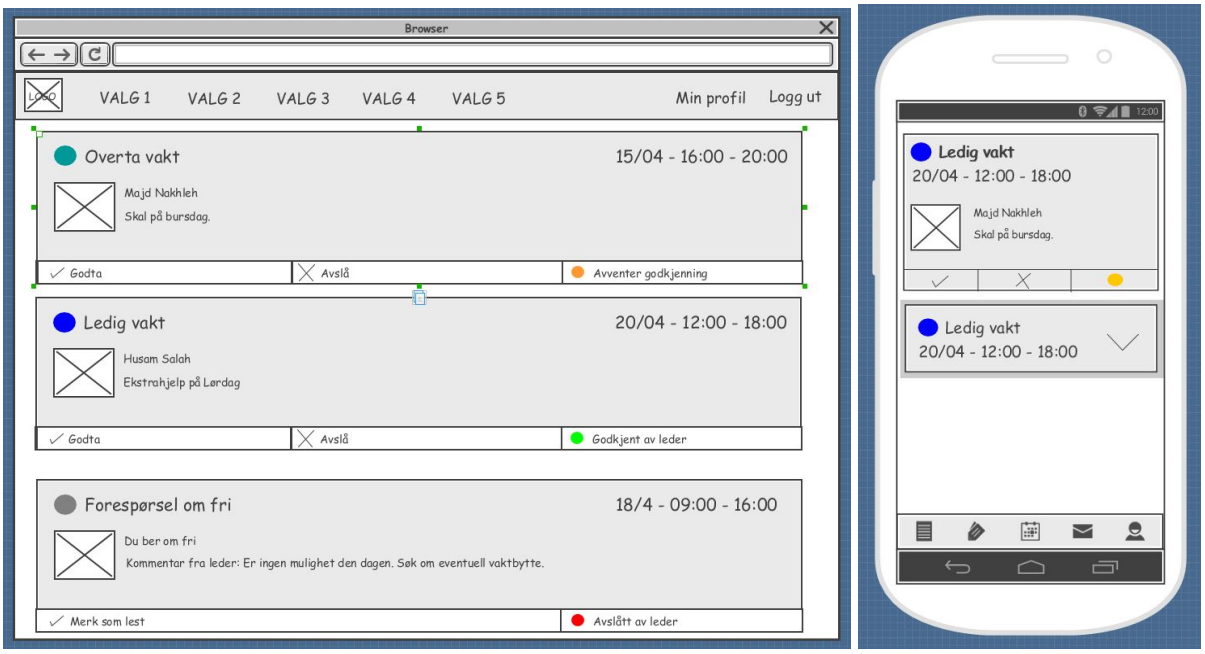


Figur 64, Wireframe forside

Skjermbilde ovenfor viser hva ansatte kan se når de er logget inn. Helt øverst for de opp en meny med ulike valg og logoen vårt som er plassert på høyresiden. Samtidig kan de klikke inn på deres profil for å legge inn eventuelle endringer, dette er plassert i høyre siden av menyen. På venstresiden får de informasjon om deres neste vakt og forespørsler, der andre ansatte kan legge inn om eventuelle vaktbytte, og se ledige vakter. Nederst i forespørsel panelet kan ansatte bruke linken som snarvei for å komme seg til forespørsel siden, der kan de administrere de forskjellig forespørsler som de har fått. På høyresiden har vi vaktplanen som viser vaktene deres uke for uke. Under vaktplanen, er det Oppslagstavle, der arbeidsgiveren legger ut nødvendige informasjon til hans ansatte.

Bilde til høyre illustrerer hvordan den samme siden skal se ut på mobile enheter. Dette er kun en forenklet versjon av datamaskin siden, her får de informasjon om når neste vakt er, antall forespørsler og oppslagstavle. Vaktlisten på de mobile enhetene vises ikke på startsiden. Nederst på siden kan vi se hvordan menyen skal se ut på mobile enheter, de fem forskjellige knappene er følgende: (Fra venstre til høyre) Hjem, Lønn, Vaktplan, Forespørsel, Profil.

Forespørsel for PC og mobile enheter

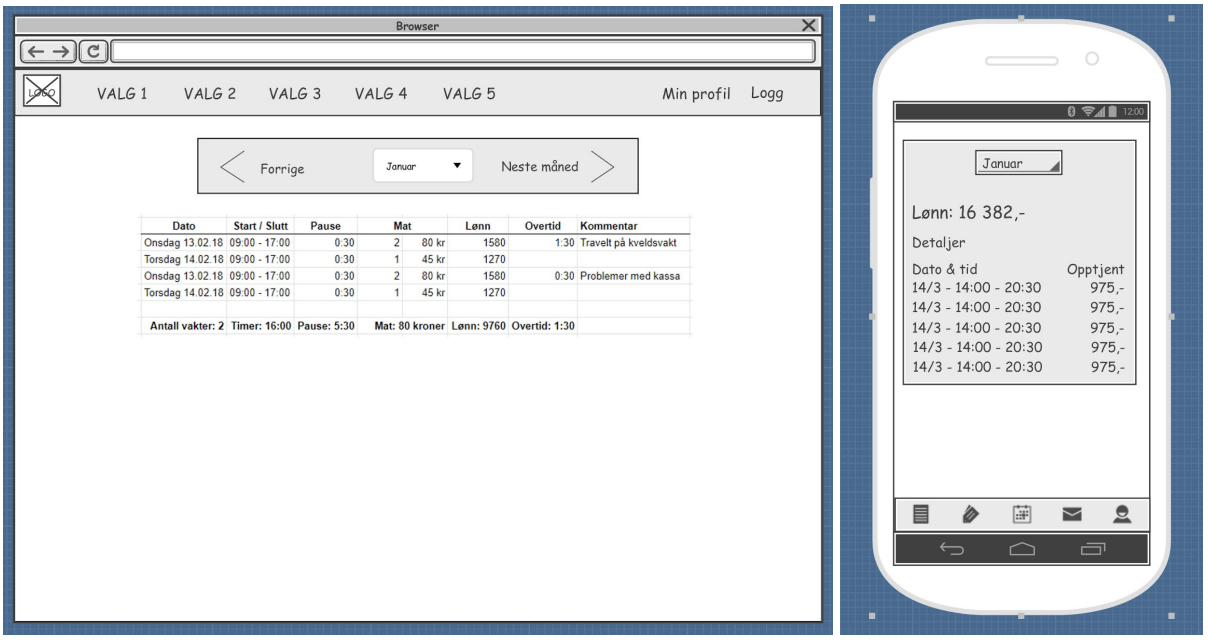


Figur 65, Wireframe forespørsler

Denne siden er for behandling av forespørsler. Disse forespørslene kan være alt fra forespørsel om ferie, fri fra en vakt, vaktbytte eller ønske om overta ledige vakter. Til venstre kan vi se hvordan vi ønsker at det skal se ut for datamaskiner. Brukere kan se forskjellige informasjon om forespørsel, og velge å godkjenne eller avslå. Helt til høyre på datamaskin versjonen kan brukere se status på forespørsel, dette kan være lønnsomt for at ansatte kan alltid vite hvor langt de er kommet i prosessen.

Til venstre er den mobile versjonen, vi kommer til å eksperimentere med at brukere skal kun se et enkel panel med informasjon og kunne utvide den for å utføre forskjellige handlinger. Dersom dette blir for vanskelig eller avansert skal vi vise hele forespørselen med alle alternativene som vist øverst på bilde til høyre.

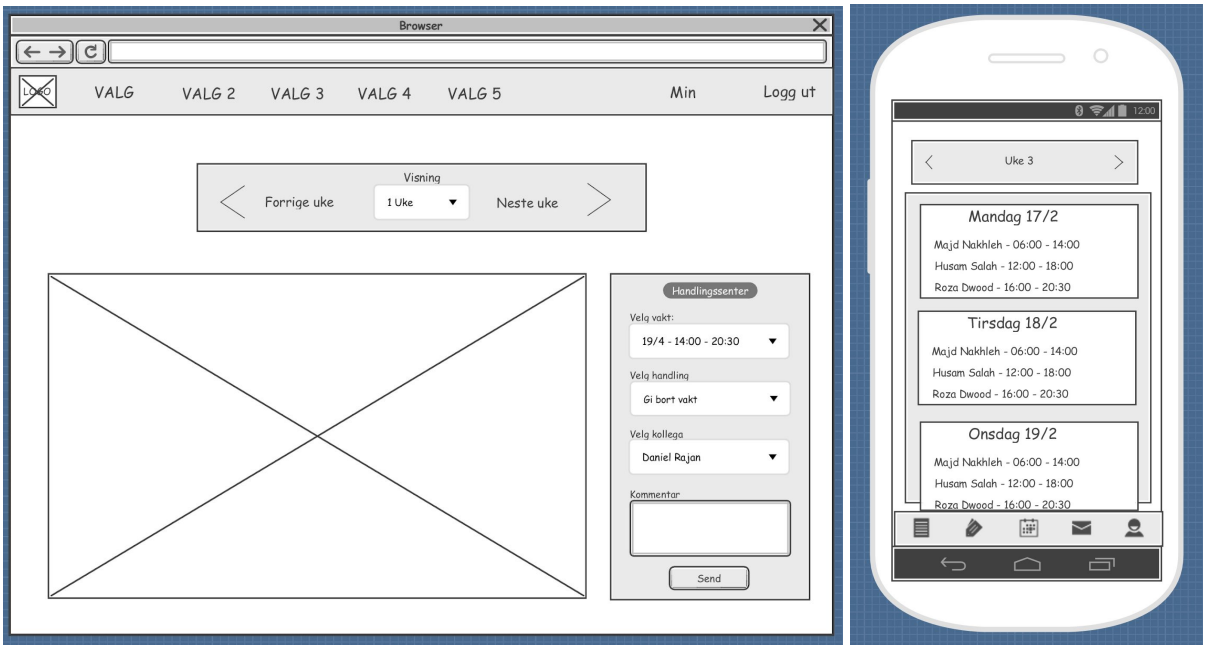
Lønningsiden for PC og mobile enheter



Figur 66, Wireframe lønnsoversikt

Denne siden skal kunne vise ansatte lønnsbeskrivelse, her kan de se opptjent lønn med en oversiktlig forklaring av hele månedslønnen. Dette er data som hentes inn fra databasen og er felles for arbeidsgiver og arbeidstaker. På denne måten kan begge partene holde god oversikt over antall arbeidstimer og korrigere dette dersom det er behov. Mobilversjonen kommer til å inneholde en mindre detaljert oversikt, det er på grunn av det blir veldig utfordrende og uoversiktlig dersom vi prøver å få inn all informasjonen på så liten skjerm. Brukere skal kunne velge selv hvilken måned de vil se lønnsbeskrivelse for, men ved første ankomst på siden vil vi vise nåværende måned.

Vaktplan for PC og mobile enheter

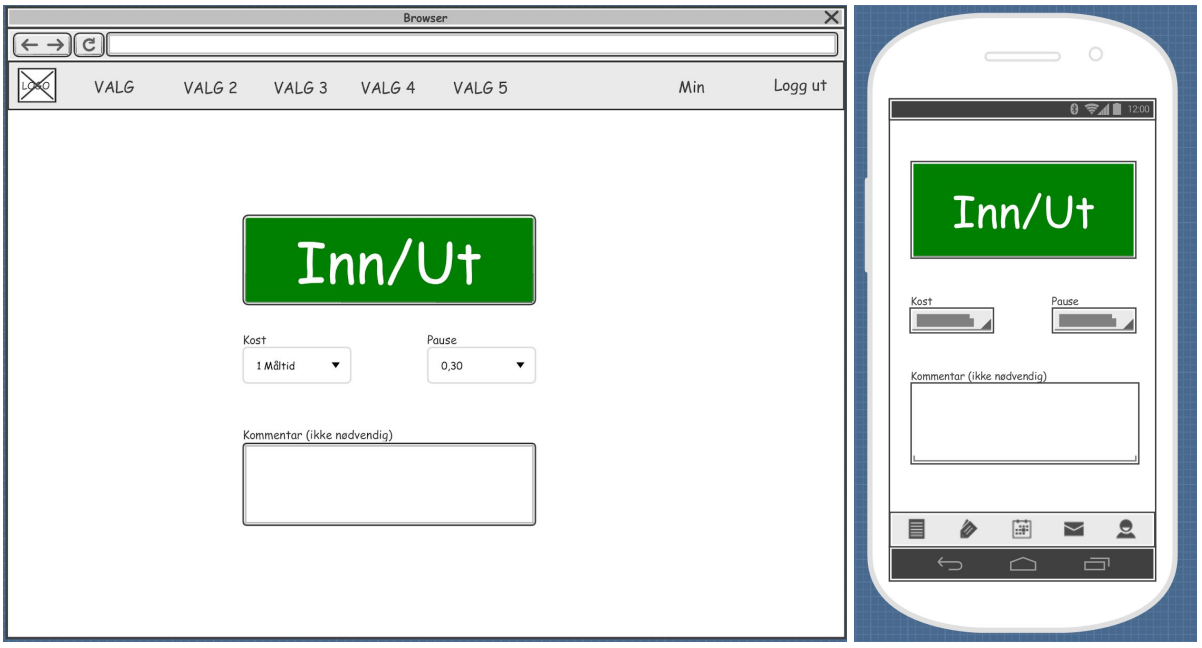


Figur 67, Wireframe vaktplan-endringer

Vaktplan siden skal kunne vise ansatte hvordan de jobber fremover. Brukeren skal kunne velge uker de ønsker å se frem i tid. Som standard kommer dette til å være en uke, vår plan er at brukere skal kunne velge mellom en, to, fire eller åtte uker frem i tid. På mobil plattform derimot viser den kun en uke frem i tid. Dette er for å holde det så oversiktlig som mulig, for mye informasjon på en liten skjerm kan føre til mindre oversiktlig nettside.

På figur 67 ovenfor ser vi et "Handlingssenter" med felter for utfylling, denne er ment for å utføre en handling på de forskjellige vaktene en medarbeider har. Her kan man velge hvilken vakt som ønskes å utføre en handling på, dette kan være alt fra å bytte vakt, søke om fri eller gi bort en vakt. Til slutt kan brukeren legge til kommentar i kommentarfeltet, og sende forespørselen og avvente godkjenning eller avslått fra administrator. Handlingssenteret skal vises både på mobil og PC enheter, brukeren som bruker mobil enhet skal kunne scrolle seg ned til handlingssenteret.

Side for inn- og ut-stempling for PC og mobile enheter



Figur 68, Wireframe innstempling

Inn- og ut-stempling skal brukes for å registrere når en ansatt kommer og forlater arbeidsplassen. Konseptet går ut på at alle brukere skal stemple seg inn når de kommer og ut stemple når de drar fra jobb, deretter kan applikasjonen bruke den dataen for å beregne antall timer ansatte har vært på jobb.

Når en ansatt stempler seg inn kan de starte med arbeidsdagen sin. Når de skal stemple seg ut for dagen, må de registrere hvor lang pause de har tatt og antall måltider som de har spist mens de var på jobb (Se figur 68). For dette skal beregnes og trekkes fra lønnen, applikasjonen på mobil og PC har kun én knapp. For å skille mellom om hva som er inn- og ut-stempling, har stemplings-knappen fargen grønn når brukeren er ikke innstemplet og kan stemple seg inn. Når stemplings-knappen er rød indikerer det at brukeren er innstemplet og kan stemple seg ut når de er ferdig for dagen.

8.6. Database

I dette vedlegget har vi samlet bilder av tabellene som finnes i databasen. For å kunne illustrere hvordan dataen som webapplikasjonen bruker ser ut, har vi valgt å inkludere testdata som ble brukt i utviklingsfasen. Under utviklingen har det oppstått tilfeller der vi har hatt behov for å endre tabeller eller legge til helt nye, dette er på grunn av at den planlagte databasen som har blitt opprettet i starten av prosjektet ikke omfatter alle funksjoner. Vi viser endringer i Figur 52, dette er en ny database-modell som illustrerer databasens komplette og siste tilstand. Endringer som har blitt gjort er som vist nedenfor, en ny tabell ble opprettet etter sprint 2, og en ny tabell ble opprettet etter sprint 3.

id_ansatt	fornavn	etternavn	adresse	postnummer	poststed	fødselsnummer	telefonnummer	stillingstype	timelønn
5	Elise	Johnsen	Hasselbakken 4	2730	Lunner	21018900000	41648501	Kokk	162
6	Daniel	Rajan	Bråtaveien 3	3511	Hønefoss	12079300000	47804722	Servitør	182
7	Roza	Dawood	Dag Hammarskjøldsvei 20	5144	Bergen	12029200000	47586951	Medarbeider	172
8	Majd	Nakhleh	Bråtaveien	3511	Hønefoss	11223300000	98736254	Kokk	162
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figur 69, Ansatt-tabell

id_bruker	epost	passord	admin
5	elise@live.no	1234	0
6	daniel@live.no	1234	0
7	roza@live.no	1234	0
8	majd@live.no	1234	1
NULL	NULL	NULL	NULL

Figur 70, Bruker-tabell

id_melding	timestamp_melding	melding	sendt_av
1	2019-02-20 10:08:00	Personalmøte neste Fredag	8
2	2019-02-22 11:50:00	Husk lønningspils på Lørdag!	8
NULL	NULL	NULL	NULL

Figur 71, Meldinger-tabell

id_vakt	id_ansatt	dato	tid_start	tid_slutt	beskrivelse
13	NULL	2019-02-21	08:00:00	16:00:00	Åpnevakt
14	NULL	2019-02-21	12:00:00	18:00:00	Kokk
15	NULL	2019-02-21	14:00:00	21:00:00	Stengevakt
16	5	2019-02-22	2019-02-22	16:00:00	Åpnevakt
17	6	2019-02-22	12:00:00	18:00:00	Kokk
18	7	2019-02-22	14:00:00	21:00:00	Stengevakt
NULL	NULL	NULL	NULL	NULL	NULL

Figur 72, Vaktplan-tabell

id_ansatt	dato_sjekketInn	tid_sjekketInn	tid_sjekketUt	timelønn	antall_timer	vaktlønn	pause	mat
5	2019-02-21	08:00:00	16:00:00	170	8	1315	00:30:00	45
5	2019-02-22	09:00:00	17:00:00	170	8	1200	00:30:00	60
6	2019-02-21	12:00:00	18:00:00	180	8	1295	00:30:00	55
7	2019-02-21	14:00:00	20:00:00	190	8	1380	00:30:00	55
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figur 73, Innsjekk-tabell

id_forespørsel	id_vakt	type_forespørsel	sendt_av	sendt_til	kommentar
3	16	1	5	6	Jeg vil selge vakta mi
4	17	2	7	8	Dette er en ledig vakt
NULL	NULL	NULL	NULL	NULL	NULL

Figur 74, Forespørsel-tabell

id_type_forespørsel	type_forespørsel
1	Selge vakt
2	Ledig vakt
3	Ønsker fri
NULL	NULL

Figur 75, Type forespørsel-tabell

Tabeller opprettet etter sprint 2:

idstatus	status
1	Ledig
2	Opptatt
3	Ønsker vakt
NULL	NULL

Figur 76, Status-tabell

Tabeller opprettet etter sprint 3:

idansatt_status	id_ansatt	id_vakt	dato	id_status
37	41	NULL	2019-04-12	2
38	41	NULL	2019-04-13	2
39	41	NULL	2019-04-14	2
40	41	37	NULL	2

Figur 77, Ansatt status-tabell

8.7. Møtereferater

Mal for møtereferat:

1. Hva slags møte referatet handler om
2. Når og hvor møtet ble holdt
3. Hvem som var til stede på møtet
4. Hva vi ønsket å snakke om og hva som ble sagt
5. Navnet på referenten

Liste over møter:

1. Møte med veileder 22.01.19
2. Møte med oppdragsgiver 19.03.19
3. Møte med veileder 27.03.19
4. Møte med oppdragsgiver 06.05.19
5. Møte med veileder 14.05.19

8.7.1. Møtereferat

1. Møte med veileder Salah Uddin Ahmed
2. Tirsdag 22.01.19 på Universitetet i Sørøst-Norge (USN) Campus Ringerike
3. Til stede: Majd, Elise og Salah heretter omtalt som “veileder” i referatet
4.
    a. Hva vi ønsket å snakke om:

- Presentere og se gjennom arbeid vi har gjort hittil, vi ønsket også å høre om vi går i riktig retning slik det planlagt frem til nå.
- Skal vi ha med milepæler? Eventuelt hvor mange og hvor store skal disse milepælene være?
- Er denne oppgave stor nok? Tanken bak dette er å få innspill dersom veileder mener at oppgaven er for liten.

- Burde vi endre på antall timer som er lagt opp til utviklings-delen i prosjektet, eller er dette godt nok med tanke på å få et fullført sluttprodukt?

b. Hva vi snakket om:

- Vi presenterte bacheloroppgaven som vi har brukt som utgangspunkt og de forskjellige fasene i systemutviklings livssyklus.
- Veileder mener at ulempen med fasene er at sensor er ikke nødvendig utvikler. Han vil at vi skal bruker strukturen til en "vanlig" bacheloroppgave, i riktig rekkefølge, og innføre systemutviklingsprosessen inn i metode.
- Vi lurer på hvor mye vi skal fokusere på produktet. Veileder mener at dette avhenger av sensor. Når sensor ikke er utvikler selv så er det mulighet for at de ikke fokuserer så mye på sluttproduktet.
- Vi lurer på om vi skal ha med alle bruksmønster som vi laget. Veileder syns vi skal lage en liste over alle og legge ved 3-4 eksempler, slik at vi kan vise at vi har lagd noen eksempler på bruksmønster.
- Legge ved møtereferater, arbeidslogger. Samme som med bruksmønster - lag en liste og legg ved noen eksempler.
- Veileder mener vi bør ha med en detaljert database-model.
- Veileder syns oppgaven virker stor nok.
- Veileder mener vi burde ha med milepæler, disse milepælene bør inneholde "Big achievements". Milepælene kan korrespondere med fasene i utviklingsprosessen.

5. Referent: Elise

8.7.2. Møtereferat

1. Møte med oppdragsgiver Husam Salah
2. Tirsdag 19.03.19 på Pasta Café Hønefoss
3. Til stede: Majd, Daniel og Husam heretter omtalt som "oppdragsgiver" i referatet
4.

a. Hva vi ønsket å snakke om:

- Vi ønsket å fremføre deler av webapplikasjonen til oppdragsgiver, formålet med dett er å gi oppdragsgive innnlikk av webapplikasjonen og få tilbakemeldinger.
- Diskutere og inkludere oppdragsgiveren i å designe oppretting av vaktplanen.

- Vi ønsker at oppdragsgiver skal gi oss mer informasjon om lønnsutbetaling, hvilken informasjon som er nødvendig og hvordan han ønsker sluttdokumentet skal se ut når det eksporteres.
- Generell tilbakemeldinger fra oppdragsgiver, og om det er noe flere systemkrav han ønsker at vi skal ta med.

b. Hva vi snakket om:

- Vi fremførte applikasjonen for oppdragsgiver som planlagt, vi fremførte hva som har blitt gjort og planene våre fremover. Oppdragsgiver mente applikasjonen så bra ut og at det vi har planlagt samsvarer med hva han hadde sett for seg.
- Vi ga begrunnelser bak valgene vi har tatt og oppdragsgiver mente mange av løsningene er gode. Oppdragsgiver uttrykket ønske om to nye funksjonelle-krav:
- Han vil gjerne ha muligheten til å godkjenne alle vakter før de blir inkludert på lønnsutbetalingen. Det krever at vi oppretter en del der han kan se alle innsjekkinger som har blitt gjort, disse kan da endres, slettes eller godkjennes av administrator.
- Arbeidstilsynet krever at alle arbeidsgivere har kontroll på hvem som er på jobb i nåværende tidspunkt og at det blir dokumentert. Per dags dato blir det brukt en bok som alle ansatte må skrive ned sitt navn i når de kommer på jobb, de noterer starttidspunkt og slutttidspunkt før de reiser hjem. Oppdragsgiver ønsker en side der han kan se hvem som er på jobb per dags dato, denne skal han kunne bruke som dokumentasjon til arbeidstilsynet.
- Dataen for å utføre disse to systemkravene finnes allerede i databasen og dermed mener vi at disse kan utføres, vi legger de til i kravlisten.
- Oppdragsgiver forklarte oss hvordan prosessen for å lage vaktlisten er per dags dato. Han var veldig åpen for at vi designer en løsning selv og hadde ingen preferanser, han forklarer at den ideelle måten er å velge en dato, deretter legge til ansatte som skal jobbe på valgt dato.
- Det siste vi pratet om var lønnsutbetaling, her ville vi ha informasjon om hele prosessen og hva som gjøres. Dette fikk vi vite mye om av oppdragsgiver og noe vi ikke hadde tatt hensyn til var overtidsbetaling og/eller helligdager. Dette er noe som er veldig viktig og som må tas opp i felleskap for å finne en god nok løsning som ikke er for avansert.

5. Referent: Majd

8.7.3. Møtereferat

1. Møte med veileder Salah Uddin Ahmed
2. Tirsdag 27.03.19 på USN Campus Ringerike
3. Til stede: Majd, Elise, Roza, Daniel og Salah heretter omtalt som "veileder" i referatet
4.

    a. Hva vi ønsket å snakke om:

    Vi ønsker å vise veileder arbeidet som er gjort frem til nå for å få tilbakemeldinger som vi kan bruke videre i arbeidet.

    b. Hva vi snakket om:

    Vi fremførte webapplikasjonen til veileder og fikk følgende tilbakemeldinger:

    - I sammenheng med eksisterende løsning for lønnsoversikt for ansatte burde det være mulig for en ansatt å se sammenlagt tid for pauser per måned.
    - Når det kommer til vaktplan så mener veileder at det vil være lettere for ansatt og daglig leder å administrere vakter dersom vi gir ansatte muligheten til å endre tilgjengelighet. Det vil se at ansatte kan velge når de ønsker å jobbe og når de ønsker å ha fri.
    - Administrator skal enkelt kunne vise mer informasjon om ansattes lønn dersom en ønsker dette. Dette skal gjøre det lettere for daglig leder å korrigere feil i lønnsslippen.

    Etter vi fikk tilbakemeldinger om webapplikasjonen, ønsket vi også tilbakemeldinger på rapporten. Vi fikk følgende tilbakemeldinger på rapporten:

    - Generelt fikk vi positive tilbakemeldinger, vi var usikre på om vi har for mange skjermdumper i rapporten men veileder mente antallet var innenfor. Veileder mener vi burde lage systemarkitektur, samtidig inkludere databasemodell ved start av prosjektet og en til på slutten av prosjektet for å vise endringer i databasen. Den originale databasemodellen burde være inkludert i designfasen og sluttresultatet burde være inkludert i systemdokumentasjonen.
    - Rapporten bør inneholde diskusjoner om følgende;
    - Hvilken forskningsprosess vi brukte.
    - Hvorfor vi valgte den teknologien vi gjorde.
    - Fordeler og ulemper mtp. sikkerhet.
    - Alternativer til valgt teknologi.

- Hva har vi lært, hvilke problemer møtte vi på og hvordan løste vi dette?
- Alternative applikasjoner som kunne bli brukt i stedet for vår webapplikasjon, fordeler og ulemper med disse alternativene.
- Utdype mer om hvorfor vi valgte Scrum metodikken.

5. Referent: Roza

8.7.4. Møtereferat

1. Møte med oppdragsgiver Husam Salah
2. Tirsdag 06.05.19 på USN campus Ringerike
3. Til stede:
   a. Majd, Daniel, Elise, Roza og Husam heretter omtalt som "oppdragsgiver" i referatet.
   b. På grunn av en misforståelse så var ikke veileder Salah til stede som planlagt.
4.
   a. Hva vi ønsket å snakke om:

- Vi ønsker å fremføre sluttproduktet til veileder og oppdragsgiver, målet er å få tilbakemelding om hva som er utført på en god måte, og hva som kunne vært forbedret.
- Er dette et god nok alternativ til det som gjøres i bedriften i dag? Vil dette effektivisere arbeidsdagen?
- Til veileder: Hvilken sikkerhetsløsningen kan vi implementere slik at systemet blir sikrere?

   b. Hva vi snakker om:

   Vi fremfører følgende sider i webapplikasjonen:

- Sjekk inn/sjekk ut.
- Lønnsutbetaling.
- Vaktplan: ønske om fri/bytte vakt.
- Opprette vakter.

   Vi har forklart hvordan oppdragsgiver ikke har mulighet til å slette ansatte permanent fra

database, og dette var noe som passet han også i og med at det finnes norske lover som sier at han må kunne dokumentere data 3 år tilbake i tid. Oppdragsgiver mener at dette er en webapplikasjon som ville forenklet både arbeidsdagen til seg selv og ansattes, samtidig som han mente at den var lett å lære og veldig brukervennlig. Tilbakemeldingen som vi fikk av oppdragsgiver var generelt sett positiv, han hadde noen endringer som vi måtte utføre; webapplikasjonen regner ut lønn fra den første dagen i måneden frem til den siste, men i bedriften til oppdragsgiver utbetales lønnen den 15 hver måned. Dette betyr at webapplikasjonen må regne fra den 15 til den 15 hver måned, noe som er mulig endre.

5. Referent: Elise

8.7.5. Møtereferat

1. Møte med veileder SalahUddin Ahmed
2. Tirsdag 14.05.19 på USN campus Ringerike
3. Tilstede: Roza, Daniel, Majd, Elise og Salah heretter omtalt som "veileder" i referatet
4.
    a. Hva vi ønsket å snakke om
    - Diskutere spørsmålet rundt sikkerhetstiltak gjort rundt applikasjonen.
    - Vise endringer gjort i applikasjonen siden siste møte
    - Fortelle om møte med oppdragsgiver 06.05.19
    - Spørre generelle spørsmål om rapporten og fremføringen

    b. Hva vi snakket om
    - Vi forteller veileder at vi ikke føler oss kompetente til å sikre applikasjonen. Han anbefaler at vi lager en generell sikkerhetsplan som en anbefaling til oppdragsgiver. Sikkerhetsplanen bør inneholde trusler, anbefalinger og personvern (GDPR).
    - Vi viser applikasjonen, spesielt de endringer som er gjort. Veileder syns det ser greit ut, men vil ikke kommentere spesifikt.

- Vi forteller om møte med oppdragsgiver.
- Vi spør etter tilbakemeldinger på rapporten. Veileder mener vi skal fokusere på å skape en rød tråd fra problem til løsning. De ulike delene av løsningen, "Team, tools, task" bør dokumenteres. Veileder ønsker at vi nummerer kapitler og figurer/tabeller i rapporten. Han understreker at vi må fokusere på rapporten og at det er viktig med grammatikk og syntaks. Rapporten leveres i WiseFlow. Koden skal ikke leveres.
- Vi spør om ulike administrative aspekter rundt fremføringen 13. juni. Veileder sier at det mest sannsynlig er én ekstern sensor for hele kullet. Han sier vi bør ha med både rapporten og produkter, altså applikasjonen, i fremføringen. Fremføringen blir vår mulighet til å "selge" applikasjonen.

5. Referent: Elise

8.8. Gruppekontrakt

Denne kontrakten regulerer arbeidet mellom medlemmene i denne bachelorgruppen og gjelder for alle som har skrevet under, fra den 25. desember til og med 13. juni. Et unntak for å bryte kontrakten er dersom alle gruppemedlemmene er enige om å gjøre endringer eller at gruppen går i oppløsning.

1. Formålet med prosjektet

1.1 Kunne besvare relevante spørsmål og samarbeide for å produsere et godt sluttprodukt og en god bacheloroppgave.
1.2 Gruppemedlemmer skal bruke all fagrelatert kompetanse som de har gått gjennom tidligere semestre i dette studiet.
1.3 Kunne utveksle forskjellig kunnskap mellom hverandre, og bistå hverandre med hjelp dersom det er behov.
1.4 Formålet innad gruppen er å få en god læringskurve gjennom dette prosjektet.
1.5 Produktet som designes skal være tilpasset etter Pasta Cafe og deres behov.

2. Orden og oppførsel

2.1 Forsinkelse må bli meldt fra til alle andre medlemmer innenfor rimelig tid.
2.2 Alle forsinkelser som er over 15 minutter telles som en anmerkning.
2.3 Alle anmerkninger rapporteres i refleksjons vedlegget.
2.4 Anmerkninger kan gis for forsinkelser, dårlig arbeidsinnsats og mobbing. All arbeid i gruppen skal foretas på en profesjonell måte.
2.5 Medlemmer må ikke forstyrre arbeidsroen for andre gruppemedlemmer.

3. Arbeidsprosess

3.1 Alle møtene starter med en gjennomgang av arbeidsprosessen.
3.2 Medlemmer har friheten til å jobbet i eget tempo så lenge de leverer tilfredsstillende resultat.
3.3 Alle møtene avsluttes med utdeling av arbeidsoppgaver som skal jobbes på egenhånd og avtale tid for neste oppmøte.
3.4 All rapportarbeid skal dokumenteres i nettbaserte Google Doc og deles umiddelbart med andre medlemmer

4. Beslutninger

4.1 Alle beslutningstagelser tas i et fellesskap.

4.2 Dersom medlemmer er uenig, er det en demokratisk avstemning.

4.3 Dersom medlemmene er fortsatt uenige, skal vi forhøre oss med veileder om innspill og råd.

4.4 J.fr Orden og oppførsel, punkt 2.1, dersom du fraværende fra et møte uten å varsle innen rimelig tid mister du din rett til å være med på beslutninger som blir avgjort på gjeldende møte.

4.5 Gruppen må lage et spesifikt delmål med rimeligere tidsestimat for innlevering.

4.6 Når en delmål er nådd, må dette dokumenteres i henhold til arbeidsprosess punkt 3.4.

4.7 Når en delmål ikke er oppnådd, må dette dokumenteres og en ny tidsramme må opprettes. Medlemmene dette gjelder kan risikere og få en anmerkning i henhold til punkt 2.4.

5. Arbeidsmiljø

5.1 En god arbeidsmiljø reflekterer på arbeidsprosessen og levere et godt resultat som gruppe.

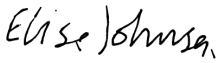
5.2 Hvis en misforståelse oppstår, så skal det tas opp i fellesskap, slik at det skapes en god dialog.

5.3 Gi gode tilbakemeldinger til hverandre, med åpen for kritikk. Kritikk skal gis på en konstruktiv måte.

5.4 Enhver i gruppen er ansvarlig for å ta opp problemer dersom det skulle oppstå noe underveis.

Hønefoss, 29. januar 2019

Roza Dawood Shakr



Elise Johnsen

Daniel Rajan

Majd Nakhleh

8.9. Brukerveiledning

Brukerveiledning for bruk av TPLAN

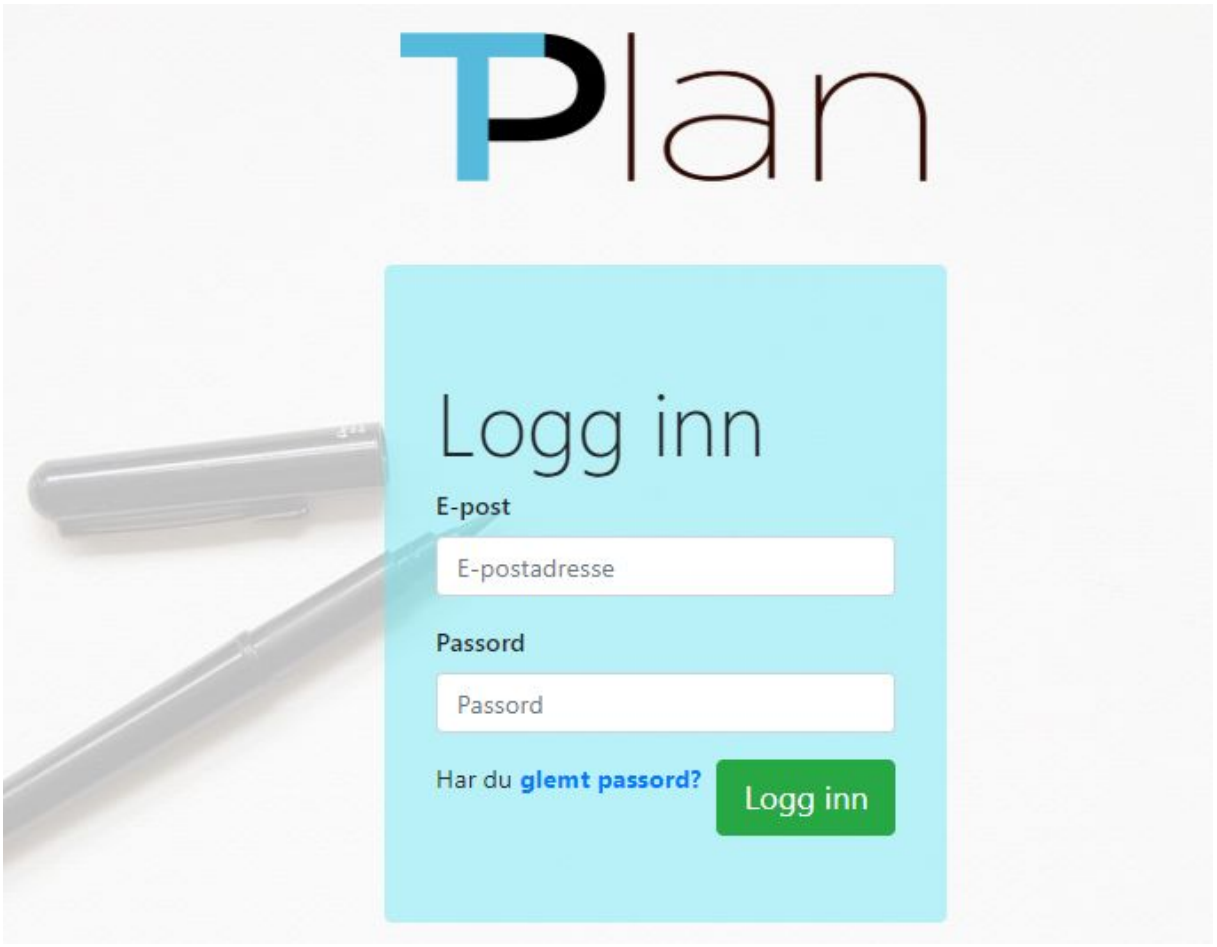
Denne brukerveiledningen er ment å være et oppslagsverk for brukere og administrator av TPlan. Den tar for seg funksjonene til applikasjonen og forteller hvordan man navigerer seg for å finne de, og hvordan de brukes.

De første funksjonene som beskrives er felles for brukere og administrator.

Deretter beskrives funksjoner for brukere, og sist administrator.

Innholdsfortegnelse

Logg inn - felles for administrator og bruker



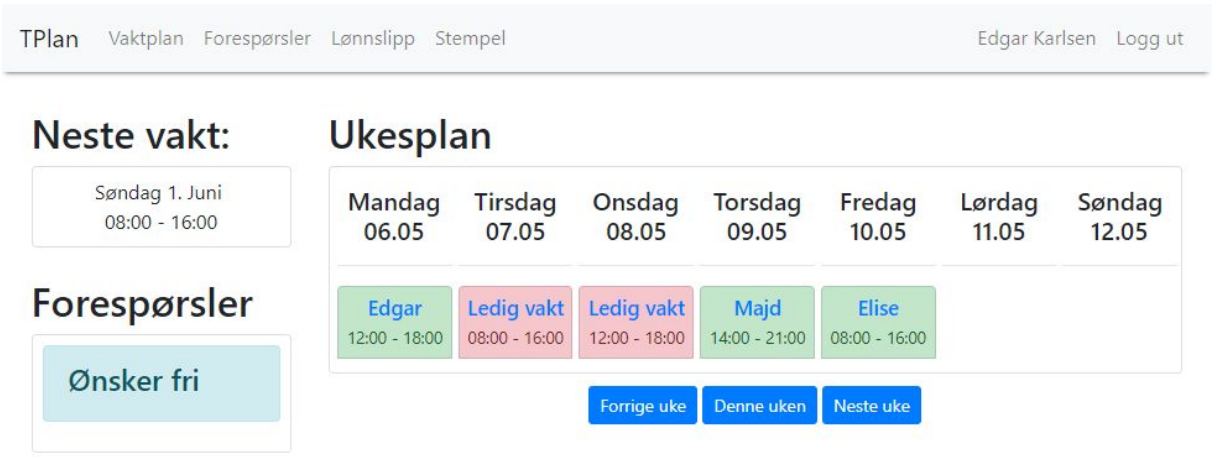
Logg inn med e-postadresse og passord som du har fått av administrator. Ved første pålogging må du endre passordet til et som bare du kjenner til.

Krav til passordet:

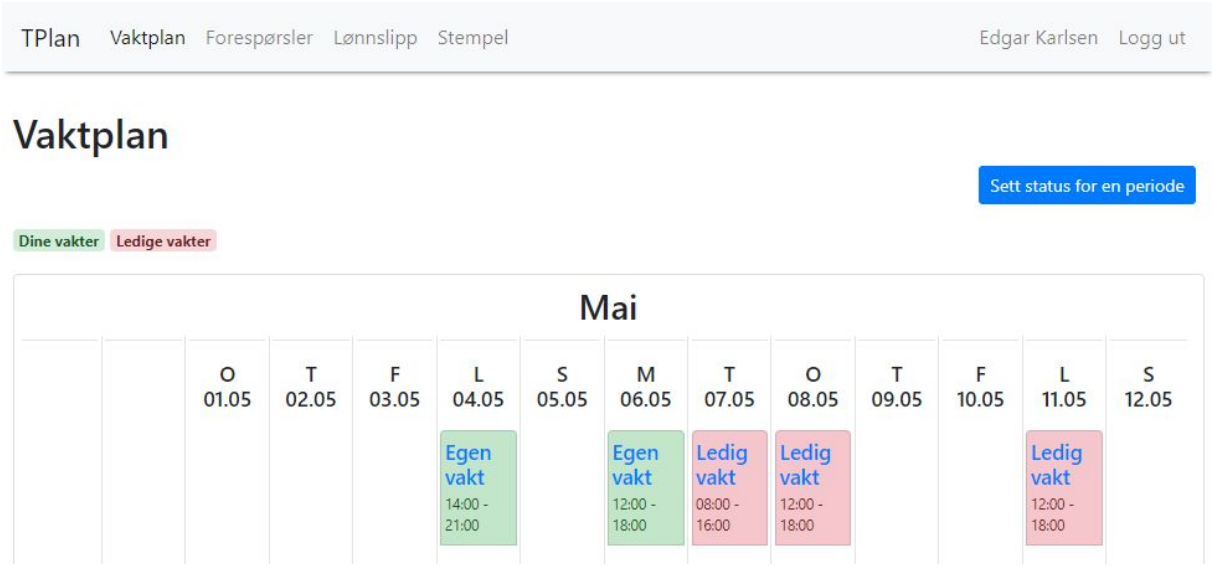
- Det må være minst 8 tegn langt
- Det må inneholde både store og små bokstaver.
- Et sikkert passord er langt. For å lage et passord som er enkelt å huske kan man benytte seg av en setning. Bruk gjerne en strofe fra en sang du liker godt, som for eksempel “Bæ, bæ, lille lam”.

Få en oversikt over vaktene dine - felles for administrator og bruker

Du kan se når din neste vakt er, og hvem som jobber denne, forrige og neste uke på forsiden.



Via vaktplanen (velge “Vaktplan” i navigasjonsbaren) kan du se egne, og ledige, vakter for de neste seks månedene.



Stemple inn / ut - felles for bruker og administrator

Når du ønsker å stemple inn på jobben går du til “Stempel” via navigasjonsbaren.

Hvis du ikke er sjekket inn vil du få tilbud om å sjekke inn. Da ser siden slik ut:

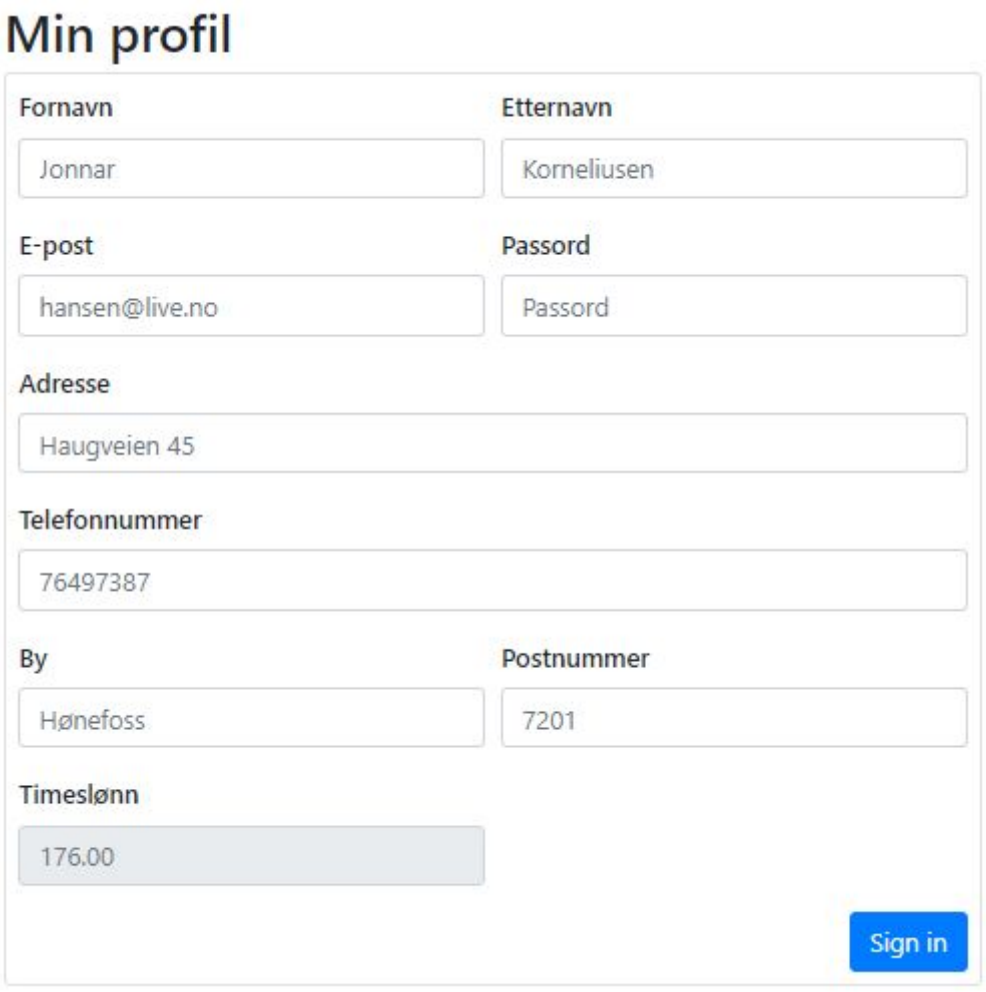
Hvis du er sjekket inn får du mulighet til å legge inn informasjon om vakten din, og å sjekke ut. Skjema for utsjekk ser slik ut:

Skriv navn på evt. måltid som ble spist i løpet av vakten, og hva den koster. Du kan legge ved kommentar til vakta som du vil at daglig leder skal se.

Pausetid velger du fra en nedtrekksmeny.

Endre egne opplysninger - felles for bruker og administrator

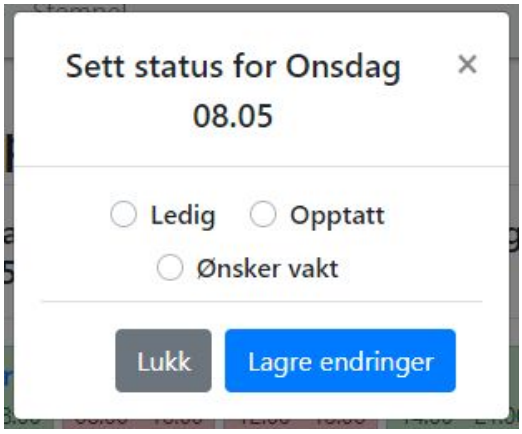
Trykk på eget navn i navigasjonsbaren for å åpne “min profil”. Her kan du endre opplysninger om deg selv.



Du kan lese av timelønna di, men denne kan ikke endres.

Sett status på vakter - bruker

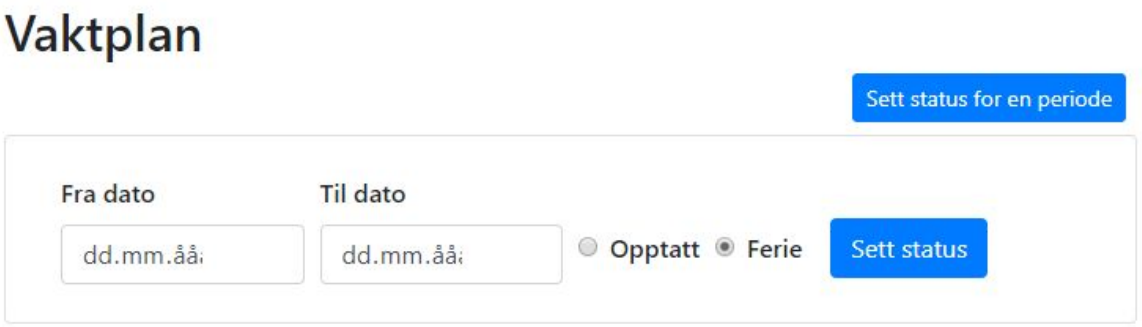
Du kan trykke på ledige vakter, både på ukeplanen og vaktplanen, for å sette status på de. Sprettoppmenyen ser slik ut:



- Velg “ledig” hvis du er ledig denne vakten, men ikke nødvendigvis ønsker å jobbe
- Velg “opptatt” hvis du er opptatt
- Velg “ønsker vakt” for å signalisere at du ønsker denne vakten

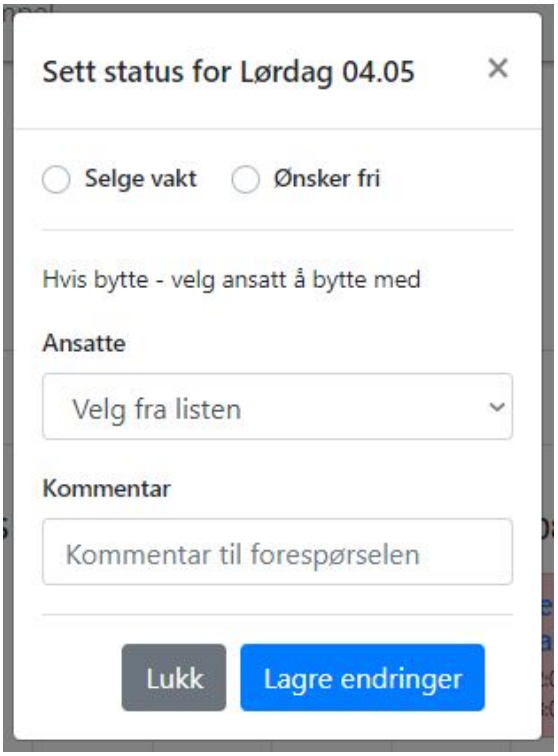
Sett status for en tidsperiode

Du kan sette status for en tidsperiode i et skjema på siden med vaktplan. Skjema er skjult når siden med vaktplan åpnes, men kommer til syne hvis du trykker på knappen “Sett status for en periode”. Her kan du sette statusen “Opptatt” og “Ferie” for en tidsperiode



Sende forespørsler - bruker

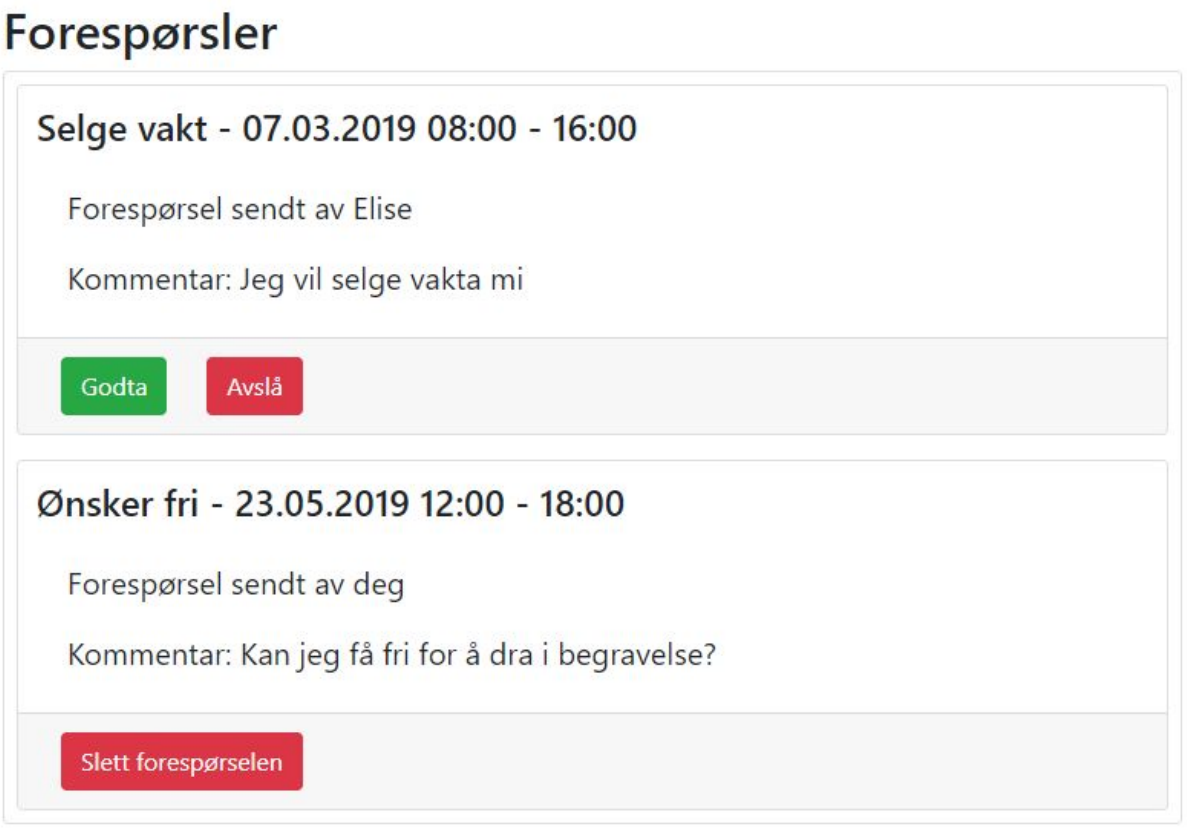
Ved å trykke på egen vakt i vaktplanen får du opp en sprettoppmeny hvor du kan velge å sende en forespørsel om enten å selge vakten din, eller å be om fri. Menyen ser slik ut:



Velg “selge vakt” eller “ønsker fri”.

Se forespørsler - bruker

På forsiden vil du få en liste over de forespørslene som er sendt til deg. Du kan trykke på en av disse for å bli sendt til siden med forespørsler. Du kan også komme til siden med forespørsler via navigasjonsbaren.
Siden med forespørsler inneholder både forespørsler som er sendt til deg, og forespørsler som du har sendt andre eller forespørsler om fri.



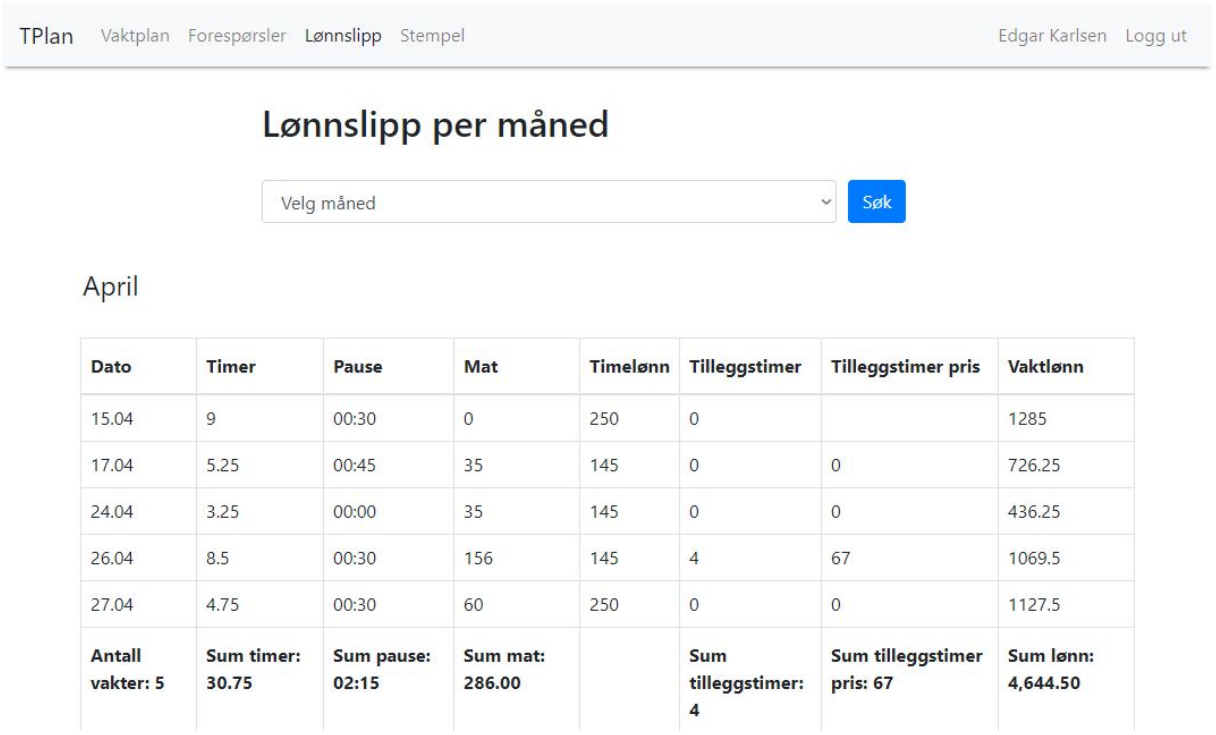
Forespørsler som er sendt til deg kan du velge å godta eller avslå.

Forespørsler som du har sendt kan du velge å slette.

Se lønnsslipp - bruker

Gå til siden med lønnsslipp via navigasjonsbaren.

På siden for lønnsslipp kan du se lønnsslipp for tidligere måneder, og en foreløpig estimert lønnsslipp for inneværende måned. Velg måned i nedtrekksmenyen og trykk på “Søk”.

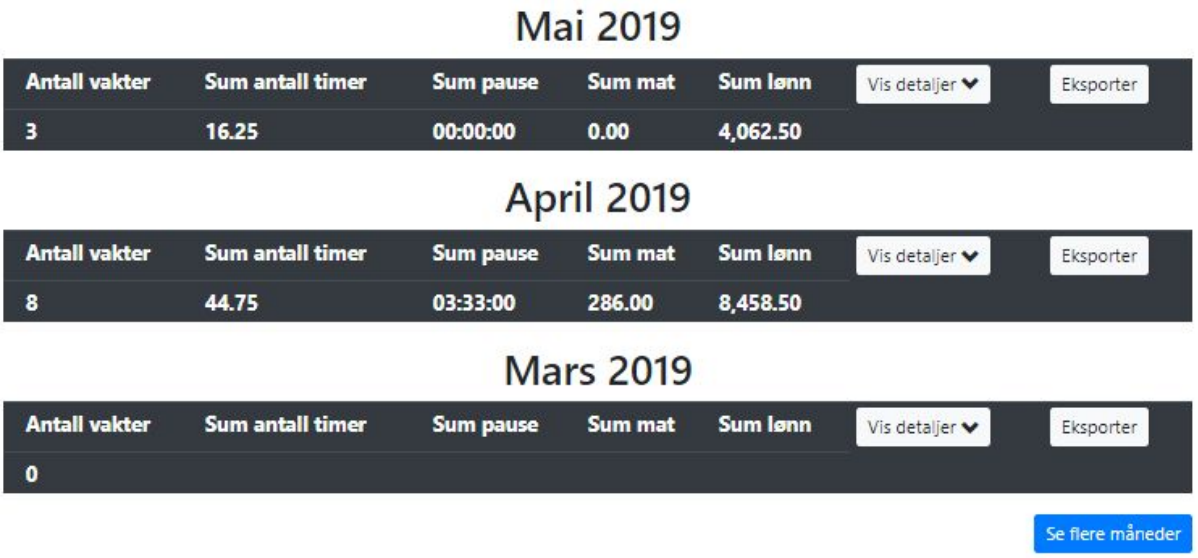


Dato	Timer	Pause	Mat	Timelønn	Tilleggstimer	Tilleggstimer pris	Vaktlønn
15.04	9	00:30	0	250	0		1285
17.04	5.25	00:45	35	145	0	0	726.25
24.04	3.25	00:00	35	145	0	0	436.25
26.04	8.5	00:30	156	145	4	67	1069.5
27.04	4.75	00:30	60	250	0	0	1127.5
Antall vakter: 5	**Sum timer: 30.75**	**Sum pause: 02:15**	**Sum mat: 286.00**		**Sum tilleggstimer: 4**	**Sum tilleggstimer pris: 67**	**Sum lønn: 4,644.50**

Se lønnsutgifter fordelt på dato / vakter - administrator

Velg “Lønn vakter” fra nedtrekksmenyen “Lønn” i navigasjonsbaren.

Det første som møter deg på siden er en oversikt over lønnsutgifter for de siste tre måneder.



Hvis du ønsker å se detaljer for hver måned, trykk på “Vis detaljer”. Da får du listet opp alle registrerte vakter for valgt måned. De er sortert kronologisk på dato.

Trykk på “Eksporter” for å laste ned en Excel-fil med lønnsdetaljer fra valgt måned.

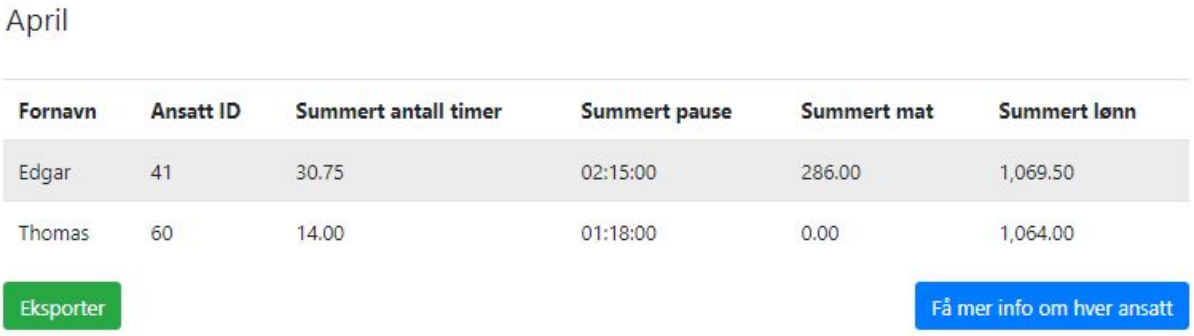
Trykk på “Se flere måneder” for å få listet opp flere måneder.

Se lønnsutgifter fordelt på ansatte - administrator

Velg “Lønn ansatte” fra nedtrekksmenyen “Lønn” i navigasjonsbaren.

Velg måned og trykk på “Søk”

En oversikt vil se slik ut:



Velg “Eksporter” for å laste ned oversikten til en Excel-fil.

Trykk på “Få mer info om hver ansatt” for å bli sendt videre til siden med lønnsslipp for hver ansatt.

Se ansattes lønnsslipper - administrator

Det er to måter å navigere til denne siden på:

- Gå via navigasjonsbaren. Velg “Lønnsslipper” i nedtrekksmenyen “Lønn”.
- Gå via “Lønn ansatte” og velg “Få mer info om hver ansatt”

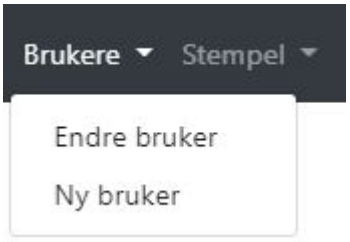
Velg måned og ansatt, og velg “Vis”.

Lønnsoversikt Edgar for April

Dato	Timer	Pause	Mat	Timelønn	Tilleggstimer	Tilleggstimer pris	Vaktlønn
26.04	8.5	00:30	156	145	4	67	1069.5
27.04	4.75	00:30	60	250	0	0	1127.5
24.04	3.25	00:00	35	145	0	0	436.25
17.04	5.25	00:45	35	145	0	0	726.25
15.04	9	00:30	0	250	0		1285
Summert	**Timer: 30.75**	**Pause: 02:15:00**	**Mat: 286.00**		**Tilleggstimer: 4**	**Tilleggstimer pris: 67**	**Lønn: 4,644.50**

Endre bruker - administrator

Gå til Endre bruker via nedtrekksmenyen Brukere i navigasjonsbaren.



Velg den ansatte som du ønsker å endre opplysninger om.

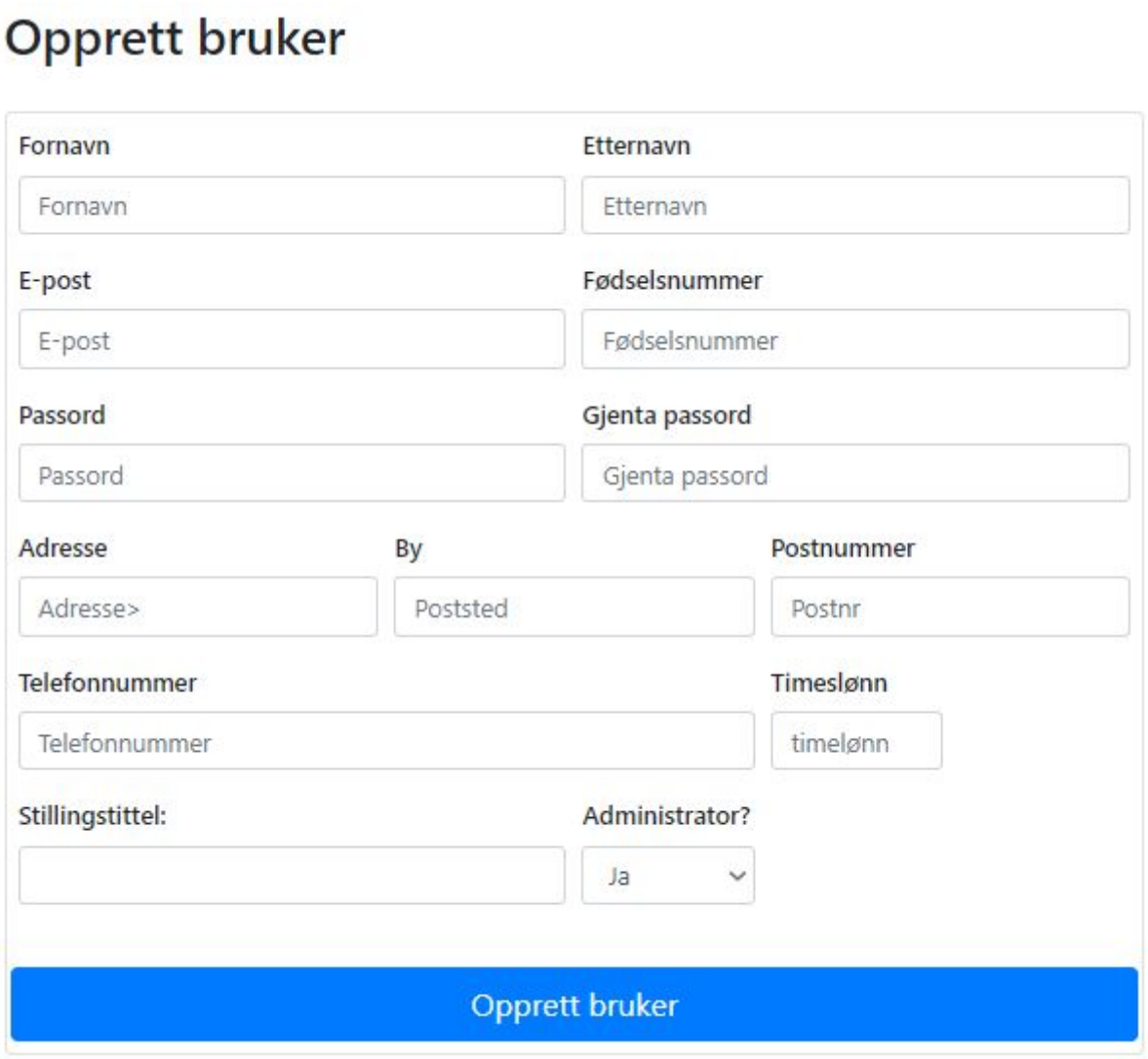
Endre opplysninger om brukeren og velg “Lagre endringer”.

Det er ikke mulig å slette en bruker, men brukere kan deaktiveres for å forhindre tilgang til applikasjonen. For å deaktivere en bruker velg den røde knappen “Deaktiver bruker”. Brukers status er opplyst på siden.

Opprett bruker - administrator

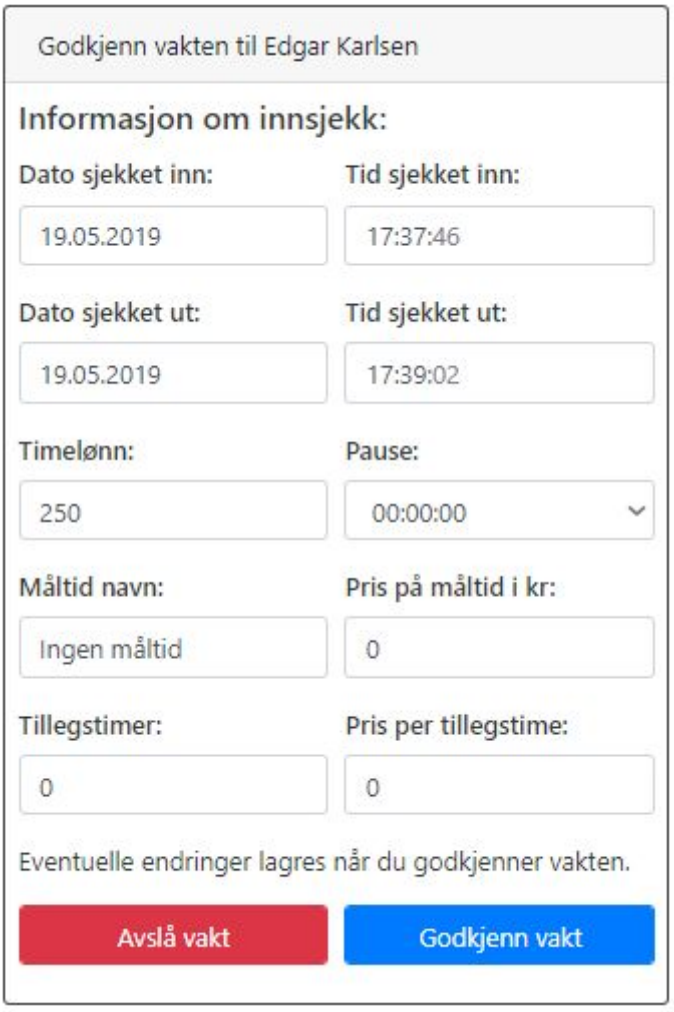
Gå til “Ny bruker” via nedtrekksmenyen Brukere i navigasjonsbaren.

Fyll inn opplysninger om brukeren og velg “Opprett bruker”



Godkjenn vakter - administrator

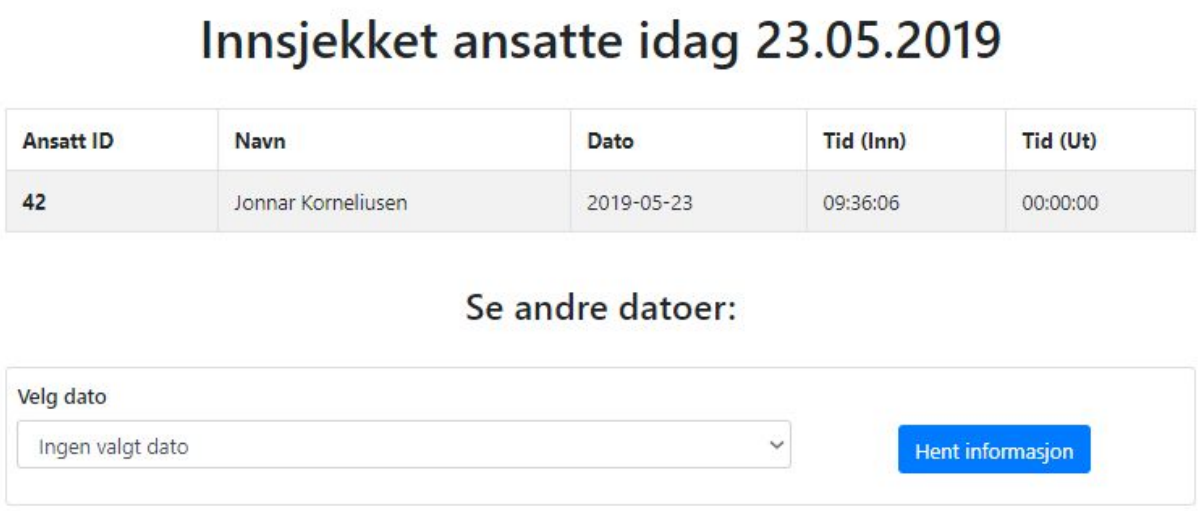
Velg “Godkjenn vakter” fra nedtrekksmenyen “Stempel” i navigasjonsbaren.
Her vil du få oversikt over vakter som er registrerte ved at ansatte har sjekket inn og ut på jobb. Her kan du endre alle opplysninger om en vakt.



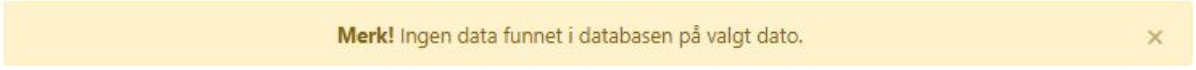
Velg “avslå vakt” for å slette den. Velg “Godkjenn vakt” for å lagre opplysningene om vakten.

Se hvem som er på jobb - administrator

Velg “Ansatte på jobb” fra nedtrekksmenyen “Stempel” i navigasjonsbaren.
Ansatte som er innsjekket vil vises automatisk i tabellen øverst på siden.
Du kan også få listet opp hvem som var innsjekket på jobb andre datoer. Velg dato fra nedtrekksmenyen og velg “Hent informasjon”.



Hvis datoen ikke inneholder innsjekker vil du få en melding om dette.



8.10. Figurliste