**Innlevering 3 – Gruppe 9**

Emil Berglund

Andreas B. Olaussen

Khalid H. Osman

Sebastian W. Thomsen

Ida K. Tollaksen

Høgskolen i Østfold

Emnekode: ITF20319

Ord: TBD

Innlevering: 27.10.2024

Innholdsfortegnelse

[Introduksjon – Tankegang og arbeidsfordeling 3](#_Toc180796601)

[Front-End – Vue & Quasar 4](#_Toc180796602)

[Kort introduksjon til Vue og Quasar: 4](#_Toc180796603)

[Fremgangsmåte og arbeidsteknikk: 4](#_Toc180796604)

[Tankegangen bak valget 5](#_Toc180796605)

[Implementasjon av Back-End i Front-End: 5](#_Toc180796606)

[Java-Backend (Hovedtittel) 7](#_Toc180796607)

[Database (Hovedtittel – Skrives av Khalid) 8](#_Toc180796608)

[Retrospekt fra forrige sprint (Alle, men spesielt Khalid.) 9](#_Toc180796609)

[Fremtidlig arbeid: 10](#_Toc180796610)

[Modeller og lenker: (Typisk skjermbilder, videoer, demo av det vi har gjort) 11](#_Toc180796611)

# Introduksjon – Tankegang og arbeidsfordeling

I denne sprinten har vi gått videre i prosessen om å utvikle vårt produkt. Fra å videre-definere produktet og designe prototype i Sprint 1, har vi nå kommet godt i gang på både Front-End og Back-End delene av produktet vårt.

Fra å jobbe mye med det samme i forrige Sprint, har vi nå splittet oss mer, og fordelt individuelle arbeidsoppgaver. Vi ble tidlig i prosessen enig om at løsningen vår burde ta løsningen som en applikasjon. Hovedsakelig fordi dette er en effektiv måte å sende varslinger på, og i tillegg en veldig enkel og intuitiv måte å se varslinger på. Denne løsningen tillater også flere utvidelser senere i løpet, hvis man ønsker det.

Ettersom Back-End delen av produktet vektlegges mer i oppgaven vi har fått, ble dette hovedprioriteten. I denne sprinten har derfor 4 av 5 medlemmer jobbet med Back-End. Som Back-End språk valgte vi Java og SQL, hvor 3 medlemmer jobbet med Java, og 1 med SQL. Dette er fordi at alle gruppemedlemmene har hatt Programmering 2 og Databasesystemer, og fordi Ole-Edvard bruker Java i forelesning.

Front-End syntes vi derimot var prikken over i-en, og var en nødvendighet for å vise hvordan vi tenker oss en fullverdig løsning. Et gruppemedlem fikk derfor arbeidsoppgaven om å skrive Front-End kode, med hovedfokus på å få til navigasjon, og fullverdig ux. Funksjonalitet som innlogging og adaptiv kode, var lav prioritet. Som språk i Front-End valget vi Vue, med komponentsystemet Quasar. Hva Vue og Quasar er, bli bedre forklart senere i besvarelsen.

Ettersom vi også har fått tilbakemeldinger på tidligere innleveringer, følte vi det også var nødvendig å se til disse. Ett gruppemedlem fikk dermed i oppgave å se over og forbedre tidligere besvarelser, slik at disse holder standarden vi ønsker til endelig innlevering.

# Front-End – Vue & Quasar

Som Front-End har vi valgt å bruke Vue og Qusar som er to helt nye kodespråk og verktøy som ingen av oss hadde vært borte i. Emil ble tipset om å bruke disse, og fikk dermed ansvaret for Front-End. Han har på egenhånd med litt opplæring fra en kompis lært seg både Vue og Quasar, for å bruke i denne oppgaven, men også for å ha kunnskapen til senere i arbeidslivet. For å ha noe å vise til det dere leser, ligger det på GitHub en video som viser og forklarer Front-End struktur og resultat. URL ligger under «Modeller og lenker».

## Kort introduksjon til Vue og Quasar:

Vue er et allsidig rammeverk for å bygge brukergrensesnitt og enkeltsideapplikasjoner (*Vue.js*, 2014). Vue bygger mye på HTML, CSS og JavaScript hvor mye syntaxen er lik. Ved å opprette et vue-prosjekt tvinges man til en viss filstruktur og visse regler ved filnavn og filreferering, men til gjengjeld får man muligheten til å koble opp mange eksterne tjenester. Eksempelvis har Vue-prosjektet gitt oss muligheten til å importere modulen «capacitor» som gir oss muligheten til å sende koden til Xcode hvor den oversettes til Swift, som er Apples kodespråk (*Capacitor*, 2024). Videre kan man laste ned en prototype på iOS enheter for videre bruk og testing.

Quasar er bygget for Vue, og er i seg selv ikke et selvstendig verktøy eller kodespråk. Quasar er et rammeverk som forenkler utvikling ved å gi brukeren tilgang til ferdigdefinerte-klasser (*Quasar,* 2020). Quasar kan brukes til flere ting, men hovedsakelig for utvikling av responsive web- og mobilapper. I vårt tilfelle, har Quasar gjort det veldig effektivt å opprette elementer som knapper med funksjonalitet, men også for stilsetting med ferdigdefinerte klasser som «q-mt-md», som gir «margin-top» med «medium» avstand.

## Fremgangsmåte og arbeidsteknikk:

Ettersom Vue og Quasar var nytt for meg (Emil) ble det en bratt læringskurve og mye arbeid for å i det hele tatt forstå kodespråket og rammeverket. Først og fremst måtte jeg initialere et Vue prosjekt og injisere Quasar rammeverket før jeg kunne begynne med kodingen.

Etter å ha opprettet et Vue prosjekt, med fungerende rammeverk kunne jeg begynne å kode. Fremgangsmåten min derifra var veldig enkel, men tok likevel lang tid grunnet mange fremmedelementer. Mitt mål under utviklingen var å kode prosjektet så likt prototypen som mulig. Jeg begynte derfor med velkomstsiden og jobbet med videre, før jeg til slutt endte på dashbord siden, og undersidene som fulgte der. Mitt fokus lå i starten kun på design, i hvert fall før jeg ble komfortabel med Vue og Quasar. Videre i løpet lærte jeg meg mer, og gikk noen ganger tilbake og endret struktur.

En ting jeg la fokus på under utviklingen var hyppige commits og pushes til GitHub. Jeg syntes dette gjør det lettere i etterkant å se tilbake til, og ikke minst for andre å forstå. Jeg utførte derfor commits så ofte som mulig, som også ga meg muligheten til å gå tilbake til tidligere versjoner. Alle originale commits vil ligge under «ux-design» på GitHub, før jeg ved ferdigstillinger av sider, har merget med develop. Main har vi blitt enige om å kun bruke til Sprint innleveringer, dere vil derfor finne alt der også.

## Tankegangen bak valget

Skriv her:

* Hvorfor Vue og Quasar
* Fordelene med valget
* Ikke HELT fremmed, ikke vanskelig å lære seg
* Åpner mange dører

## Implementasjon av Back-End i Front-End:

Tanken videre, for det endelige produktet er å implementere Java-Back-End og SQL-Back-End sammen med Front-End. Hvordan dette gjøres, vet vi foreløpig ikke, og det har derfor blitt noen midlertidige løsninger for å illustrere hvordan løsningen skulle sett ut.

Slik Front-End fungerer ved innlevering, er det satt opp en Firebase som i samarbeid med google lar en bruker registrere seg, og logge inn. Ved å logge inn med google, får vi som utviklere tilgang til; fornavn, etternavn og e-post ved første øyekast. Dette kan videre brukes til brukerkontroll og brukerhåndtering. Slik det er nå, kan man derfor ikke registrere seg med tradisjonelle skjemaer. Her er det da tiltenkt at man skal kunne bruke disse skjemaene og sende det til SQL-databasen som tar vare på dette.

Hovedpunktet i appen er å varsle brukeren når den kommer hjem om det lønner seg å lade eller ikke. Slik det er nå, snakker ikke front-end koden sammen med Java koden, som henter denne informasjonen ved bruk av API. Tanken her, er å visualisere og hente nødvendig informasjon som Java får fra API. Dette kan hentes ved at Java skriver til en lesbar JSON fil, som deretter Front-End kan bruke og videreformidle brukeren.

Videre må vi ha en måte å varsle brukeren på, om betingelsene rundt posisjon og strømpriser er sanne. Det vi ønsker her, er at Java skal håndtere varslingen, og at Front-End skal implementere det. Java skal derfor sjekke strømprisen og brukerens posisjon og avgjøre hva brukeren bør gjøre, før resultatet kan leses av Front-End. Slik det er nå, er dette ikke koblet til Front-End.

# Java-Backend (Hovedtittel)

Strømpriser (Andreas)

* Konto-håndtering (Sebastian)
* Varslinger (Ida)
* GPS (Khalid)

Tips fra Canvas:

* Nevn/oppsummer typiske funksjoner eller funksjonalitet
* Legg vekt på strukturen dere har brukt i filene deres. Typisk løst koblet kode og Test Driven Development. Forklar hvorfor dere har gjort slik det er.
* Nevn at dere har forskjellige bruksscenarioer, inkludert feilsituasjoner for ALT.

# Database (Hovedtittel – Skrives av Khalid)

# Retrospekt fra forrige sprint (Alle, men spesielt Khalid.)

* Khalid: Skriv hvilke endringer du har gjort
* Alle andre: Oppsummer hva vi har gjort uten å kopiere innledningen, forklar hvordan vi har arbeidet.

**Endring av tidligere dokument**

Etter første utkast av tidligere dokumenter mottok vi en rekke ting vi kunne forbedre på for å gjøre dokumentet mer komplett og strukturert. Vi måtte blant annet legge til kapittelnummering, slik at sensor enkelt kan navigere og forstå innholdet. Vi manglet også to brukerhistorier som vi måtte legge til basert på scenarioet. Til slutt måtte vi legge til litt mer spesifikke situasjoner i scenarioet.

Ut ifra tilbakemeldingene vi fikk på scenarioet så vi at vi manglet en del detaljer, som kunne gi et mer realistisk bilde i hvordan appen vår ville fungere i praksis. Etter endringer beskriver scenarioet nå bedre hvordan en typisk bruker ville brukt produktet. Endringene vil la til fokuserte på å lage et bilde av «Hvordan brukeren bruker systemet når han kommer hjem etter en lang dag på jobb» og «Hvordan systemet hjelper brukeren med å ta smarte ladebeslutninger uten å måtte bruke tid på å sjekke priser selv». Disse endringene gir leseren en bedre forståelse av produktet.

Den andre tilbakemeldingen vi fikk gikk ut på å legge til kapittelnummering. Dette skulle bidra til å gjøre prosjektet mer oversiktlig. Derfor la vi til en innholdsfortegnelse. Dette gjør det rett og slett lettere for leser å finne fram til de ulike delene av dokumentet.

Den siste endringen vi gjorde var å legge til brukerhistorie. Dette var noe vi ikke hadde i det første utkastet. Vi valgte derfor å legge til to spesifikke brukerhistorier for å illustrere hvordan og hvorfor bruker ville brukt produktet. Begge disse bruker historiene er relevante til det vi allerede har og handler om de brukerne vi brukte i både Personasene og scenarioene.

# Fremtidlig arbeid:

* Forklar hva vi skal fokusere på fremover
* Fokus på struktur i Java

# Modeller og lenker: (Typisk skjermbilder, videoer, demo av det vi har gjort)

* Emil: GitHub sti til video.
* Ta eventuelt eget initiativ her, om dere vil inkludere noe over.

Litteratur:

*Vue.js*. (2014). Vuejs.org. <https://vuejs.org/>

*Quasar Framework - Build high-performance VueJS user interfaces in record time*. (2020). Quasar Framework. <https://quasar.dev/>

*Capacitor by Ionic - Cross-platform apps with web technology*. (2024). Capacitor. <https://capacitorjs.com/>