Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Kristian Kampenhøy
Simon Lilleeng

Utvikling av mobilapplikasjon for Fixrate AS

Bacheloroppgave i Bachelor i ingeniørfag, data
Veileder: Nils Tesdal

Mai 2020

NTNU
Kunnskap for en bedre verden

Kristian Kampenhøy
Simon Lilleeng

Utvikling av mobilapplikasjon for Fixrate AS

Bacheloroppgave i Bachelor i ingeniørfag, data
Veileder: Nils Tesdal
Mai 2020

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU

Kunnskap for en bedre verden

Forord

Bacheloroppgaven, for dataingeniør vår 2020, er utstedt og gjennomført i samarbeid med Fixrate AS, for Institutt for datateknologi og informatikk (IDI) ved Norges teknisk-naturvitenskapelige universitet (NTNU).

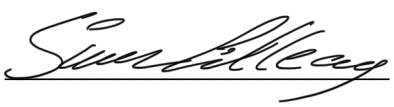
Å utvikle en mobilapplikasjon som et tilskudd til Fixrates allerede eksisterende webapplikasjon har vært en spennende og lærerik oppgave. Oppgaven ble allerede utformet høsten 2019 i faget TDAT3022 Systemutviklingsprosjekt og vi begge har siden da lært utrolig mye når det kommer til ny teknologi og utvikling. Vi tok kontakt med Nils Tesdal, ansatt utvikler hos Fixrate og foreleser på dataingeniør hos NTNU, og lurte på om de hadde en oppgave vi kunne jobbe med. Vi fikk tilbud om å utvikle en mobilapplikasjon for deres kunder og, om ønskelig, mulighet for å videreføre prosjektet til en bacheloroppgave. På forkant av prosjektstart hadde ingen av oss erfaring med å utvikle applikasjoner for mobile plattformer. Gjennom tiden vår hos Fixrate har vi lært utrolig mye nytt som vi kan ta med oss videre i våre roller som systemutviklere.

Vi vil takke Bjørn Oddvar Landsem, oppgavestiller og kundeansvarlig i Fixrate for tilbakemeldinger og for å ha vært tilgjengelig når vi har ønsket. Vi vil også takke Simen Bjørkhaug for all teknisk hjelp vi har fått og generelt alle utviklerne hos Fixrate for å ha vært behjelpelige og delt deres kunnskap med oss.

Vi vil også rette en takk til Olava Faksdal for all hjelpen med utformingen og utluking av rene trykkfeil.

Helt til slutt vil vi takke Nils Tesdal, vår veileder på NTNU, for oppfølging og rådgivning gjennom bacheloroppgaven.

Kristian Kampenhøy



Simon Lilleeng

Oppgavetekst

Fixrate har det siste året hatt en økning i kunder og har gjennom tilbakemeldinger mottatt ønsker om en applikasjon for å utføre enkle oppgaver for de som til dels er ute på reise. FixrateApp skal være en mobilapplikasjon for alle registrerte brukere av Fixrate sin eksisterende webløsning. I dagens webløsning finnes det ingen dedikert liste over hva en bruker trenger å gjøre. Brukeren må logge inn i nettleser for deretter å undersøke hvilke oppgaver han trenger å gjøre, disse er spredt litt rundt omkring på nettstedet. Fixrate ønsker derfor at alle pushvarsler som mottas på telefon legges i en egen liste som representerer gjøremål for den enkelte bruker. FixrateApp skal derfor være en løsning for å effektivisere enkle oppgaver, gi en ordnet oversikt over disse å effektivisere prosessen ved store bankinnskudd for brukerne [1].

Sammendrag

I Bacheloroppgaven ble det utviklet en mobilapplikasjon med formål å effektivisere en bestillingsprosess for de eksisterende kundene som bruker Fixrate sitt nettsted. Problemstillingen vi har valgt har vært aktuell under hele utviklingsperioden av applikasjonen.

Oppgaven ble løst ved å utvikle en mobilapplikasjon og er ment for å tilgjengeliggjøre Fixrate sine tjenester for kunder uten direkte tilgang til en datamaskin. Gjennom applikasjonen vil brukerne motta pushvarsler når det det er behov for deres oppmerksomhet i forbindelse med en bestilling. Gjennom applikasjonen kan brukeren se markedet til Fixrate, utføre enkle oppgaver som å signere og bekrefte avtaler, starte oppsigelser og bestille nye innskudd. Hver bruker har sin egen liste hvor alle pushvarsler som krever deres oppmerksomhet legges inn, slik kan de hele tiden ha full kontroll over hva de trenger å gjøre i systemet.

Applikasjonen er integrert i et eksisterende system som til daglig er i bruk og problemstillingen er derfor knyttet direkte mot dette. Vi har under utviklingen benyttet oss av Fixrate sine systemer og databaser for å integrere applikasjonen på best mulig måte slik at den kan gi kundene en følelse av merverdi og nytte.

Innholdsfortegnelse

Figurer

Akronymer og forkortelser

Android	Mobilt operativsystem, basert på en modifisert versjon av Linux-kjernen
Backend	Den delen av programvaren som ligger nærmest databasen og der dataene er lagret
DOM	Datamodell og et API for HTML- eller XML-dokumenter
Events	Handling eller hendelse som kan håndteres av programvaren
FaceID	Ansiktsgjenkjenningssystem designet
Frontend	Et grensesnitt, den delen som ligger nærmest brukeren
IDE	Et integrert utviklingsmiljø
iFrame	HTML komponent som viser viser innhold fra gitt URL i en ramme
iOS	Operativsystem for Apple's iPhone
Java/ JavaScript	Programmeringsspråk
JWT	JSON Web Token
NIBOR3M	Den renten norske banker er villig til å låne hverandre penger til i tre måneder
NTNU	Natur teknisk-naturvitenskapelige universitet
SAML	XML-basert standard for utveksling av autentiserings- og autorisasjonsdata mellom sikre domener
Scrumban	Metode for smidig utvikling
SSL-sertifikat	Et digitalt sertifikat
TouchID	Elektronisk funksjon for gjenkjenning av fingeravtrykk

UIKit	Rammeverk som definerer kjernekomponenter til et iOS-program
WebView	React-Native komponent som viser innhold fra gitt URL i en ramme
Widget	Miniprogram, moduler
Wireframes, mockups	Enkle tegninger, skisser for en visuell guide av utseende på produkt
XML	Extensible Markup Language

1 Introduksjon og relevans

De fleste i befolkningen i dag bærer rundt på en smarttelefon. Smarttelefonene har stor datakraft og kan derfor brukes til nesten akkurat det samme som en datamaskin. Derfor har de fleste store nettsteder sine egne dedikerte mobilapplikasjoner for brukere med smarttelefon. Slik kan nettstedene alltid være i kontakt med brukerne sine og i mange tilfeller oppnå større synlighet. Brukere kan få beskjed direkte på telefonen om noe nytt forekommer på nettstedet og alltid holde seg oppdatert. Resultatet av dette er at brukere benytter seg av tjenestene til de som tilbyr applikasjoner i større grad og brukerne sitter igjen med en følelse av merverdi.

Fixrate har registrert et ønske og behov fra sine kunder om dette og ønsket derfor at vi skulle starte utviklingen av applikasjonen som skal brukes av deres kunder en gang i fremtiden.

1.1 Arbeidsgivers behov

1.1.1 Fixrate

Fixrate tilbyr kundene sine en markedsplass for store bankinnskudd på nett. De gjør det enklere for organisasjoner med god likviditet å finne de beste tilbudene på innskuddsprodukter hos norske banker. En av visjonene til Fixrate er at ting skal være enklest mulig ved å la kundene slippe å tenke på dokumenter og tunge prosesser.

Tidligere var det ikke vanlig at store kunder hadde innskudd i en liten bank, men vha. Fixrates markedsplass kan bankene bli enda mer synligere for kunder. Fixrate har gjennom sin tjeneste skapt konkurranse blant bankene om å tilby de beste innskuddsrentene, og samtidig forenklet prosessen for organisasjoner å finne de beste innskuddsrentene bankene kan tilby.

1.1.1 Behovene og hvorfor

Når en kunde velger å sette inn et innskudd gjennom en annonse på Fixrates markedsplass startes det en prosess mellom banken som la ut annonsen og organisasjonen som skal sette inn penger i banken. Primært er det to kundetyper i Fixrate - bankansatte og innskytere. De som har rollen bankansatt kan legge ut annonser på nettstedet med en gitt rente, oppgjørsdato, minste- og største innskuddsbeløp på vegne av banken. De som bestiller og setter inn penger gjennom annonsene på markedsplassen til Fixrate kalles for innskytere. Innskytere tilhører organisasjoner som ønsker å sette inn penger.

Ved et innskudd startes en prosess i systemet til Fixrate. Bank og innskyter blir nødt til å utveksle og signere dokumenter til hverandre før avtalen og innskuddet er fullført. Alle dokumenter genereres og signeres i systemet og alt forekommer digitalt på nettstedet til Fixrate.

Denne prosessen ønsker Fixrate å effektivisere ved å ha en applikasjon som tilskudd til sitt allerede eksisterende system slik at det er mulig å varsle kundene sine når det er deres tur å gjøre en handling i innskuddsprosessen. Kundeansvarlig i Fixrate, Bjørn Oddvar Landsem, har gjennom sin kundekontakt fått tilbakemeldinger og ønsker fra kundene om en enklere måte å aksessere systemet når man bruker mobiltelefon. Fixrate hadde selv et ønske om å vise innhold fra nettsiden direkte i applikasjonen, slik at endringer på design av sentralt innhold på nettstedet ville vises i applikasjonen uten å måtte endre kildekoden til applikasjonen.

1.2 Tidligere arbeid

Vi hadde tidligere i faget TDAT3022 Systemutviklingsprosjekt høsten 2019 startet innledningen til bacheloroppgaven. Høstens prosjekt var mest en forberedelse til bacheloroppgaven hvor hovedfokus var å bli kjent med de ulike teknologiene vi trengte for å utvikle en mobilapplikasjon. Siden applikasjonen skal integreres i et eksisterende system var det derfor mye arbeid i å sette seg inn i hvordan ting hang sammen i dagens løsning. Dette var en viktig del for å kartlegge hvordan applikasjonen skulle utvikles, hva som kunne bli gjenbrukt og hvilken ny funksjonalitet det var behov for.

Resultatene fra prosjektet utført på høsten ble derfor et godt grunnlag for overgangen til bacheloroppgaven. I høstprosjektet ble det implementert pushvarsler, realisert en markedsplass med samme funksjonalitet som på nettstedet til Fixrate. Det ble i slutten også utviklet et ønsket utkast av en side hvor brukeren kunne se nødvendige gjøremål for prosessen ved bestilling av innskudd. Løsningen rundt innloggingen til applikasjonen ble derfor ikke fullstendig. Under utviklingen oppstod det et problem hvor brukeren ikke ble autentisert etter BankID-innlogging og tatt videre inn til applikasjonen på telefoner med iOS [2].

1.3 Problemstilling

På bakgrunn av systemutviklingen har vi kommet frem til følgende problemstilling:

Hvordan lage et nytt system for mobile plattformer som både baserer seg på og sameksisterer med et allerede eksisterende system?

- *Hvordan sikre brukerne en sikker innlogging med gjenbruk av løsning fra eksisterende system til nytt system?*
- *Hvilke designprinsipper må praktiseres for å ikke introdusere nye designelementer i systemet?*

1.4 Rapportens struktur

Rapporten starter med en introduksjon av gruppens tidligere arbeid og oppgavestillers behov. I neste kapittel tar vi for oss teori og konsepter knyttet til systemutvikling, sikkerheten og designprinsipper. Etter det vil vi beskrive hvilke teknologier og metoder som ble tatt i bruk under systemutviklingen, hvordan arbeidet ble gjennomført og tanker rundt hvordan designet skal bli. Resultatdelen viser hvilke resultater vi har oppnådd. Utviklingen deles her inn i tre: Vitenskapelige resultater, knyttet til produkt og design, Ingeniørfaglig resultater som er knyttet til mål satt i visjonsdokumentet og administrative resultater som er knyttet til fremdriften i systemutviklingen. I diskusjonsdelen går vi inn på hvorfor bestemte valg ble tatt, konsekvenser av dem og hvordan resultatet ble på bakgrunn av disse. I konklusjonen tar vi for oss hva vi har kommet frem til gjennom systemutviklingen satt mot problemstillingen samt. videre arbeid.

2 Teori

Dette kapittelet tar for seg teorien bak oppgaven. Her forklares konsepter og verktøy i systemutvikling, teorien rundt sikkerhet brukt i applikasjon og den eksisterende løsningen til Fixrate. Deretter ulike designprinsipper som ble brukt under utformingen av applikasjonen.

2.1 Kontinuerlig Integrasjon

Kontinuerlig integrasjon er en systemutviklingsmetode hvor utviklere regelmessig lagrer sin kode til et sentralt lagringsområde. Ved bruk av kontinuerlig integrasjon vil den nye koden som skal sjekkes inn til lagringsområdet, bli tatt gjennom flere integrasjon steg før den lagres på lagringsområdet. Ved innsjekk bygges koden og det vil bli gjennomført et sett av tester. Dette er for å forsikre at den nye koden som implementeres fungerer godt sammen med eksisterende kode. Hvis en av testene feiler, stoppes byggingen, og vedkommende som har sjekket inn kode blir varslet og kan deretter finne årsaken og gjøre endringer slik at byggingen kan bli godkjent. Målet er å hele tiden ha en kjørbar tilstand på systemet, og ved kontinuerlig integrasjon oppdage feil tidligere i utviklingsprosessen [3].

2.2 Samhandlingsverktøy

Vi har benyttet oss av flere samhandlingsverktøy gjennom denne bacheloroppgaven. Disse verktøyene gjør det enklere for grupper å samarbeide på felles dokumenter, ha videosamtaler og mer over internett.

2.3 Versjonskontroll

Under utvikling av systemer blir det ofte brukt versjonskontrollsystemer. Med slike systemer kan en enkelt lagre og oppdatere filer. For hver ny oppdatering eller lagring som skjer mot systemet, vil det bli laget en ny versjon. En kan enkelt gå tilbake til tidligere versjoner for å hente gamle filer eller filer som har blitt slettet eller mistet i nyere versjoner. Et versjonskontrollsystem kan også brukes for å lokalisere feil ved et system hvis det eksisterer tidligere versjoner av et system som har fungert, men ikke lenger gjør det. Man kan gå tilbake til den siste versjonen som fungerte og deretter feilsøke hvilke nye endringer som kan ha forårsaket at systemet ikke lenger fungerer [4].

2.4 Designprinsipper

For å oppnå et brukervennlig og godt design i applikasjonen har vi tatt i bruk flere designprinsipper gjennom utviklingen. Av disse har vi sett mest på tre prinsipper som vi mener er viktige for utviklingen av en applikasjon som i dette tilfellet skal være et tilskudd til et eksisterende nettsted. Ved å følge disse prinsippene kan man enklere oppnå gode og effektive brukergrensesnitt. Her er det prøvd til best mulig grad å holde utviklingen innenfor prinsippene affordance, consistency og feedback.

Affordance handler om at brukeren skal kunne forstå hvordan de forskjellige elementene fungerer og hva som kan ageres på kun ved å se dem. På noen elementer kan det for eksempel brukes flat utforming for å visualisere at de ikke er klikkbare, mens de med skyggelegging oppfattes som klikkbare.

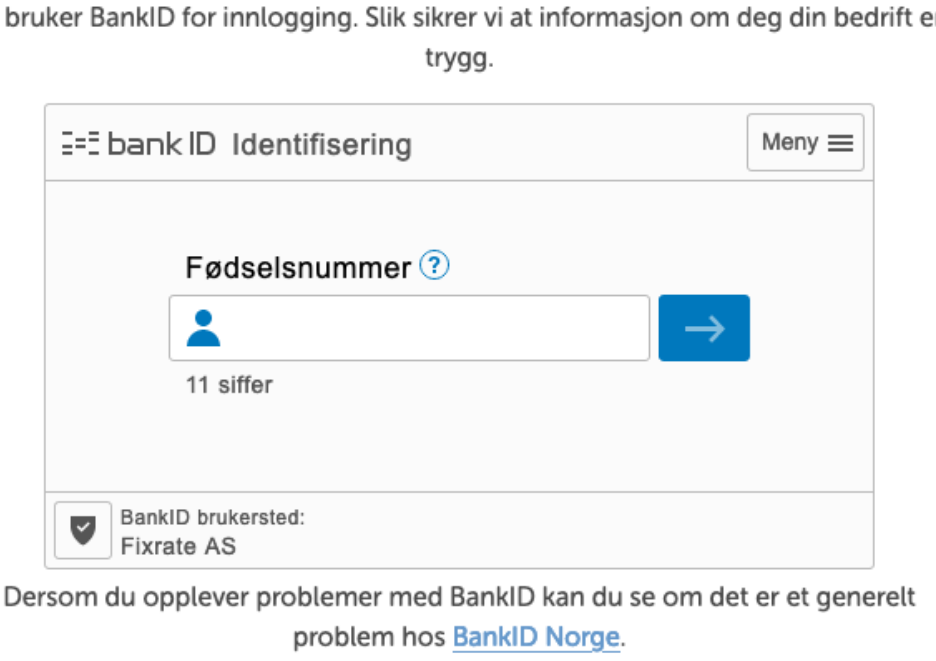
Consistency i applikasjonen er et designprinsipp som setter utseende i fokus. Her skal utformingen av design være konsistent internt i systemet og liknende i forhold til den eksterne tjenesten. Den eksterne tjenesten i dette tilfellet er Fixrates nettsted. Consistency i applikasjoner gjør det enklere å ta i bruk nye applikasjoner, fordi man designer systemet slik at det stimulerer brukerens tilegnede handlingsmønster fra erfaring av bruk av andre applikasjoner

Feedback tar for seg tilbakemeldinger brukeren får når han agerer på noe i systemet og kan være i form av visualisering, lyd eller begge deler. Feedback er ment til å gi klare tilbakemeldinger til brukeren på hva som er gjort og hva som foregår i systemet [5].

2.5 Sikkerhet Innlogging med Bank-, Touch- og FaceID

2.5.1 BankID

Det eksisterende systemet til Fixrate bruker en løsning tilbudt av Signicat [34] med trygg innlogging ved bruk av BankID. Fixrate viser en iFrame med et utsnitt av BankID-innlogging på sin nettside. Nedenfor er et skjermbilde som viser dagens innlogging på nettstedet.



Figur 1: Dagens innlogging på Fixrates nettsted

Ved gyldig innlogging vil nettleseren motta en SAML-response fra Signicats tjeneste. SAML er en XML-basert standard for utveksling av autentiserings- og autorisasjonsdata mellom sikre domener. I dette tilfellet er det en utveksling mellom (Identity Provider) Signicat [34] og (Service Provider) Fixrate.

All kommunikasjon bruker et forsterket to-veis SSL-sertifikat, IP-filtrering og hver bruker har sitt eget unike brukernavn og passord for å oppnå helt sikker kommunikasjon [6]. Enkelt forklart fungerer innloggingen slik:

1. Brukeren prøver å logge inn på Fixrate sine tjenester i nettleser ved å bruke BankID.
2. Fixrate genererer en SAML-request med kredentialene som ble gitt
3. Nettleseren sender SAML-requesten til Signicats tjeneste
4. Signicat parser SAML-requesten, autentiserer brukeren og genererer en SAML-response og sender tilbake til nettleseren.
5. Nettleseren sender SAML-responsen til Fixrate for verifisering.
6. Hvis verifisering blir godkjent får brukeren en egen cookie og session-id som brukes til å aksessere de forskjellige tjenestene i nettstedet.

SAML blir sett på som en standard protokoll for håndtering av identitet og verifisering mellom tjenester og deres kunder. SAML brukes i dag av de fleste tjenestene som tilbyr BankID-innlogging. Nedenfor er et skjermbilde som viser hvordan en SAML-response ser ut og link til hvor den skal videresendes i Fixrates system for verifisering [7].

Figur 2: Skjermbilde av SAML-response

Som nevnt i punktene ovenfor vil brukeren være autentisert hvis verifiseringen til Fixrate ble godkjent og brukeren er deretter innlogget.

2.5.2 Biometrisk innlogging

Biometrisk innlogging tillater brukeren å logge inn med maskinvare som er tilgjengelig på telefonen og består enten av fingerskanning eller ansiktsgjenkjenning. Ved å ta i bruk biometrisk maskinvare på telefonen legger man til et ekstra lag med sikkerhet. Når man setter opp maskinvaren på telefonen er det påkrevd at du også setter opp en kode, slik at maskinvaren har noe å falle tilbake på dersom den ikke klarer å autentisere deg. Under oppsetningen er det nødvendig å skanne finger eller ansikt flere ganger slik at maskinvaren får en best mulig måling. Det genereres deretter en hemmelig nøkkel ut fra disse målingene som kun operativsystemet har tilgang til. Denne nøkkelen brukes senere til å autentisere deg som bruker når du skanner ansiktet eller fingeren din.

Denne teknologien har gitt bedrifter og applikasjonsutviklere mulighet til å bruke denne maskinvaren for å kunne gjøre innloggingen sin både lettere og mer brukervennlig.

2.6 Brukertesting

Brukertesting er en effektiv metode for å få tilbakemeldinger fra brukeren om hva som fungerer godt og hva som ikke fungerer med løsningen som man presenterer [9]. Ved brukertesting er det vesentlig at en tenker på hvilken målgruppe løsningen er ment for og deretter kjører testene mot denne gruppen. Veldig ofte så er det enkelt å friskmelde løsningen som man selv har laget når det blir testet på andre involverte i prosjektet.

For å få best mulig tilbakemelding på den løsningen man ønsker å teste må man tenke over testgruppen - at de er representative brukere, og at de kan løse reelle oppgaver knyttet mot løsningen som skal testes. Ved brukertesting er det viktig at brukeren får konkrete og relevante oppgaver som skal løses, og det er viktig å reflektere over hvorfor man gjennomfører brukertesten. Er man i tvil om brukeren klarer å gjennomføre testen, må det lages et scenario som får brukeren til å enkelt forstå hva som skal gjøres.

Når man først har tatt seg tid til å gjennomføre brukertester er det derfor viktig å forsøke å lære mest mulig om hvordan brukeren oppfatter løsningen som en presenterer. En måte er å be brukeren snakke høyt når testen gjennomføres, og at brukeren forteller hva han eller hun tenker underveis når oppgaven skal løses. Personen som leder testene, er også nødt til å ta notater gjennom hele testen. Oppfølgingsspørsmål må være nøytrale slik at brukeren ikke blir ledet i den retning som er ønskelig for personen som leder brukertesten [10].

3 Valg av teknologi og metode

Dette kapittelet tar for seg valg av teknologi og metode. Her informeres det om hvilke teknologier og metoder som ble benyttet i systemutviklingen og hvorfor disse ble valgt. Vi tar også for oss metodikken som er brukt og rollefordelingen innad i gruppen. Mange av teknologiene som er tatt i bruk er enten felles, eller bygd på de samme teknologiene som brukes i Fixrate sitt system, hvor vi bruker de samme teknologiene fordi det er ønskelig at applikasjonen skal kunne kommunisere godt med det eksterne systemet. Hvordan systemet skal vedlikeholdes har også vært en sentral tanke for valgene vi har tatt.

3.1 Systemutvikling

3.1.1 React-Native

For utviklingen av applikasjonen har vi valgt å bruke rammeverket React-Native som er en teknologi basert på React. Grunnen til at vi har valgt å bruke dette rammeverket, er fordi det eksisterende systemet bruker React, og React-Native fungerer godt på ulike mobile plattformer. Siden react også er et veldig populært rammeverk blant utviklere i dag er det en veldig relevant erfaring å ta med seg videre i arbeidslivet. React-native er laget spesifikt for utvikling av mobilapplikasjoner, og mange store applikasjoner på markedet i dag som for eksempel bildedelingstjenesten Instagram [35] og kommunikasjonstjenesten Discord [36] bruker dette rammeverket. Teknologien er et åpent kildekode-rammeverk laget av Facebook [37], og baserer seg på de samme prinsippene som blir brukt i React [11]. I react bygger man små, gjenbrukbare komponenter, som knyttes sammen til noe større. Den store forskjellen mellom React og React-Native, er at der React knyttes mot DOMen, knyttes React-Native mot UIKit hos Ios og tilsvarende biblioteker hos Android. Med bruk av dette rammeverket kan mye kode gjenbrukes på tvers av plattformene som effektiviserer utviklingen av applikasjonen. [12]

3.1.2 Balsamiq Mockups

For å komme raskt i gang med å visualisere hvordan applikasjonen skulle se ut ble det lagd wireframes til oppgavestiller. Balsasmiq Mockups [13] er et veldig enkelt verktøy for nettopp dette og bærer preg av at de ser håndtegnede ut. Balsamiq Mockups har også den fordelen å være veldig enkelt og intuitivt å bruke. Dette er en av hovedårsakene til at vi valgte å bruke Balsamiq for å lage wireframes. Det er enkelt å lage detaljerte wireframes og en slipper å bruke mye tid på å sette seg inn i de mer avanserte verktøyene.

3.1.3 Expo

Expo er et rammeverk og en plattform for universelle React applikasjoner. Rammeverket inneholder et sett av verktøy som gjør det enkelt å utvikle native applikasjoner med programmeringsspråket JavaScript [14]. Expo client er en applikasjon tilgjengelig fra distribusjonsplattformer og nettbutikker for applikasjoner. Gjennom utviklingen kan man enkelt kjøre applikasjonen ved hjelp av klienten uten å måtte bygge prosjektet for hver gang det gjøres endringer.

Expo tilbyr også flere andre funksjonaliteter, som hot-reload, og et API for håndtering av pushvarsler på både Android og iOS [15]. Sistnevnte ble av stor betydning for utviklingen av FixrateApp for å effektivisere prosessen med bestilling av innskudd hos bank, men samtidig unngå å måtte legge til ekstra teknologier som enten fungerer for flere plattformer, eller som fungerer for hver enkelt av dem. Mye av grunnen for at vi har valgt å bruke Expo er fordi rammeverket gjør mye av det komplekse med å lage en mobilapplikasjon for oss, slik at vi heller kan legge vekt på det som skal utvikles.

3.1.4 Discord

Digitalt kommunikasjonsverktøy som ble mye brukt under utviklingen for å forenkle kommunikasjon over nett.

3.1.5 Microsoft Teams

Digitalt samhandlingsverktøy som tilbyr video- og telefonmøter. Teams ble brukt gjennom hele utviklingen av applikasjonen siden oppgavestiller ikke hadde kontor i samme lokale som vi jobbet i. Derfor ble det som regel avholdt videomøter for å vise progresjon i utviklingen.

3.1.6 RabbitMQ

For å håndtere de eventene som kommer fra markedsplassen, bruker Fixrate en RabbitMQ server, som er en såkalt message broker [16]. Den fungerer slik at når det kommer inn events til serveren, legges de i en kø helt til den er klar for å behandles. Måten vi bruker RabbitMQ serveren på er at vårt system kobler oss på serveren til Fixrate. Så når Fixrate lager eventer med sin server, lytter vår server på samme port og kan motta eventene som ligger klar i køen. Vi bruker denne teknologien for å oppnå pushvarsling i applikasjonen. Når det kommer inn events på RabbitMQ serveren om at det har blitt lagt ut en ny annonse på markedsplassen, fanger vår server opp dette og utfører handlinger, som for eksempel å finne enhetene som skal motta varslingen om ny annonse på markedsplassen.

3.1.7 React Redux

React redux er et veldig populært frontend-rammeverk. Rammeverket håndterer tilstanden til all data i applikasjonen [17]. Siden applikasjonen er avhengig av samme data flere steder, var vi nødt til å ha en global aksess til data. I stedet for å sende samme data gjennom flere komponenter i applikasjonen kom vi fram til rammeverket React redux som håndterer dette for oss. Slik kan vi hele tiden ha kontroll på all data i applikasjonen, når og hvorfor det endrer seg.

3.1.8 Git

Git er et moderne versjonskontroll-system som brukes i stor grad av de aller fleste systemutviklere, og er tilgjengelig på stort sett alle operativsystemer og IDEs [18]. Siden vi er flere som skriver kildekoden til applikasjonen er vi avhengig av et godt versjonskontrollsystem for å alltid ha samme kildekode og distribuere endringer til hverandre gjennom utviklingsperioden. Git gir også brukerne full tilgang til kildekodens historie og dermed muligheten til å se gjennom hele historien til koden. Dette åpner også muligheten til å "gå tilbake i tid" og oppdage hvor og når feil blir introdusert gjennom hele utviklingen.

3.1.9 GitLab

Siden vi er flere på gruppen er vi nødt til å lagre all kildekode eksternt slik at alle har tilgang til den. Gitlab er en git-basert plattform som implementerer mange verktøy for systemutviklingen og håndtering av store prosjekter. Gitlab muliggjør lagring av kildekoden eksternt med bruk av versjonskontrollsystemet Git. Gitlab gir brukerne full oversikt over kildekoden og man kan se gjennom alle filer og får et grafisk grensesnitt over utviklingshistorien til kildekoden [18]. Under utviklingen ønsket vi også at koden skulle testes kontinuerlig slik at vi enkelt kunne fange opp endringer som introduserer feil i kildekoden. Når ny kode blir lagt til kjøres det derfor tester, og vi får tilbakemelding om noen av disse feiler. Siden vi har tidligere erfaringer med kontinuerlig integrasjon gjennom GitLab ble det derfor naturlig for oss å velge denne plattformen.

3.1.10 Trello

Vi ønsket å ha arbeidsoppgaver oppført på en tavle, der vi kategoriserer oppgavene fra open, sprint, todo, doing, done og review. Vi hadde ikke tilgang til tavle og en fast plass å oppholde oss på under utviklingen, så vi brukte derfor den virtuelle tavle-tjenesten Trello. Trello er et oversiktlig samarbeidsverktøy som lar andre medlemmer lage lister, kort, oppgaver, og tildele oppgaver til andre [19]. Vi valgte å bruke dette verktøyet siden det ga oss oversikt over hva som skal gjøres, hva som jobbes med og vi kunne sette opp tavlen som ønsket fra begynnelsen av.

3.1.11 Google Drive

Google Drive er en fillagrings- og synkroniseringstjeneste som muliggjør samarbeid i dokumenter i sanntid, lagring av filer og tilgjengelighet for alle med tilgang. Siden oppgaven består av mye dokumentasjon, er det derfor nødvendig med en slik tjeneste. Siden vi har brukt Google Drive veldig mye de siste årene var det helt naturlig å velge denne tjenesten under utviklingen.

3.1.12 Jest

Jest er et rammeverk i JavaScript som brukes for testing og har stort fokus på enkelhet og fungerer i prosjekter som bruker en rekke teknologier. Av de teknologiene vi har valgt å bruke i prosjektet fungerer Jest på Babel, Node og React [20]. I prosjektet er Jest i hovedsak brukt for å forsikre oss om at funksjoner i backend fungerer som tenkt å at de henter ut riktig data fra databasen. Testene kjøres hver gang ny kode blir lagt til gjennom git for å forsikre oss om at vi ikke har introdusert nye feil.

3.1.13 Enzyme

Enzyme er et frontend-rammeverk i JavaScript som brukes for testing av React-komponenter [21].

3.1.14 JUnit

Junit er et testrammeverk som brukes for enhetstesting i Java [22]. JUnit er et rammeverk som vi er kjent med fra tidligere av i studieløpet, og som har blitt brukt i tidligere systemutviklingsprosjekter. Rammeverket lar oss teste spesifikke metoder for seg selv, på den måten kan vi forsikre oss om at metoden gjør akkurat det den skal.

3.1.15 Babel

Siden vi skriver med programmeringsspråket JavaScript, valgte vi å bruke Babel som et verktøy for å konvertere nyere versjoner av JavaScript bakoverkompatibel med eldre versjoner. For eksempel så er lambdauttrykk støttet med eldre versjoner, og det er der Babel hjelper til med å formatere koden over til vanlige funksjoner ved kompilering [23].

3.1.16 Node js

Node js er et JavaScript-rammeverk som gjør at man kan lage en asynkron server-tjeneste som kan håndtere forespørsler fra flere klienter samtidig. Dette betyr at serveren mottar en forespørsel fra klienten, starter behandlingen og mens den venter på data som skal sendes tilbake kan den behandle nye forespørsler fra andre klienter [24]. Node js har i prosjektet blitt brukt for å lage en web-server slik at applikasjonen kan hente eller lagre data i databasen. Lagring av ny data som ikke eksisterer i Fixrates systemer i dag har vært nødvendig for å muliggjøre use-case som biometrisk innlogging og pushvarsling. Det blir derfor nødvendig for Fixrate å implementere denne løsningen for å få applikasjonen til å sameksistere med

deres webplattform. Node js ble valgt som rammeverk siden gruppen har tidligere erfaringer med det.

3.1.17 Express js

Express er et minimalistisk web-rammeverk for Node js som skrives i JavaScript. Express js har derfor blitt brukt som en direkte følge av at vi bruker Node js. Express brukes for å bygge og starte webserveren og muliggjør bruken av et restful API. Express hjelper med å håndtere routes, endepunktene til serveren, og hvordan server-requests skal håndteres [25].

3.1.18 MySQL

MySQL er et håndteringssystem for databaser. For å håndtere, legge til, endre og hente ut nødvendig data i applikasjonen har vi valgt å bruke MySQL. MySQL tar for seg databaser som relasjoner, hvor objekter lagres i databasen i sine egne tabeller og knyttes mot andre tabeller ved hjelp av nøkkelpar i stedet for å lagre all data i en stor tabell [26]. MySQL ble valgt som teknologi siden det er gode biblioteker som kan brukes i Node js for å skrive MySQL-setninger for lagring/henting av data i databasen.

3.1.19 Docker

Docker er et verktøy som er laget for å gjøre det enklere å kjøre applikasjoner ved å bruke noe som heter containere. Containere tillater utviklere å pakke en hel applikasjon med alle biblioteker, rammeverk og avhengigheter i en eneste pakke. Man kan deretter laste opp containeren til sin konto på nettstedet til docker, også kalt Docker Hub. Deretter kan man laste ned containeren fra hvor som helst og starte opp applikasjonen som er pakket inn i containeren [27]. Dette gjør det derfor veldig enkelt å dele store applikasjoner, og de kan derfor startes ved å kjøre en enkel kommando i terminalen. Docker har blitt brukt siden vi var nødt til å sette opp en egen server som applikasjonen kan aksessere. Vi har deretter bygd en docker-container av kildekoden til node-serveren vår og lastet ned på vår server hvor den står og kjører.

3.1.20 Kubernetes

Som regel er store applikasjoner avhengig av flere deler. En webapplikasjon trenger i veldig mange tilfeller en frontend, backend, database og kanskje en RabbitMQ tjeneste. Som nevnt over beskrev vi Docker hvor man kan gjøre hver del om til en container. I store systemer blir det gjerne flere containere som inneholder sin egen lille del i et stort system. Kubernetes er laget for å holde styr på alle disse containerne. Disse pakkes sammen til et “cluster” som inneholder flere containere. Kubernetes holder styr på alle disse slik at om en krasjer så starter den opp selv. Det gjør det derfor veldig enkelt å distribuere store systemer. Det kan for eksempel være greit for utviklere å ha hele “clusteret” på maskinen slik at de enkelt kan kjøre hele systemet lokalt på sin maskin for å teste nye ting [28].

Grunnen for at vi har brukt kubernetes under utviklingen er for å ha mulighet til å lytte til markedsplassen til Fixrate samt. skaffe oss tilgang til deres database.

3.1.21 DigitalOcean - Droplets

DigitalOcean er et amerikansk selskap som tilbyr kundene sine, mot betaling, egne sky-servere på ett klikk. Disse kalles for Droplets og man velger selv hvilket distribusjonssystem hver Droplet skal kjøre [29]. I vårt tilfelle har vi valgt å kjøre Ubuntu [38] på serveren vår. Den kan aksesseres og konfigureres vha. SSH-innlogging. Ubuntu-serveren har blitt konfigurert med en egen MySQL-database og har også Docker installert. Kildekoden til vår NodeJS-server har blitt bygd til et docker-image og lastet ned på Ubuntu-serveren. Docker-imaget står og kjører hele tiden og vi har eksponert en port slik at vi kan aksessere node-serveren som kjøres utenfra.

NodeJS har ingen problemer med å kjøre lokalt på vår maskin, problemet oppstår derimot når vi kjører applikasjonen på telefonen. Da vil vi ikke kunne hente ut data fra en server som kjører lokalt på maskinen. Derfor var vi nødt til å få en server opp på nett slik at vi kunne hente eller lagre data under kjøring.

3.1.22 Biometrisk innlogging

Apple sier at sannsynligheten for at en tilfeldig person kan se på din iPhone og låse den opp med FaceID er ca 1 til 1 000 000 når du har ett ansikt registrert. Videre bruker iPhone teknologi som tar i bruk sofistikerte nevrale nettverk for å beskytte mot forsøk på å lure maskinvaren ved hjelp av masker eller andre teknikker. FaceID registrerer også når brukeren fokuserer på telefonen, om øyne er åpne og om oppmerksomheten din er rettet mot telefonen [8]. Fixrate sitt system brukes av mange kunder og det er derfor viktig at det er en viss sikkerhet rundt innlogging på applikasjonen.

3.2 Arbeids- og rollefordeling

Begge studentene har jobbet sammen ved flere anledninger under studietiden og er derfor godt kjent med hverandres kunnskap og ferdigheter. Begge ble enige om å arbeide sammen da de valgte faget TDAT3022 Systemutviklingsprosjekt for å utfordre seg selv både faglig og personlig. Begge har den samme bakgrunnen som tidligere tømrere og startet studiet med ingen forkunnskaper om programmering. Begge har i løpet av studietiden fått stor interesse for programmering og trives generelt i både front- og backend.

Simon Lilleeng har hatt hovedansvaret for backend-løsningen. Dette ble naturlig å gjøre siden han jobbet med en del av applikasjonen som hadde behov for tilgang til en backend.

Kristian Kampenhøy har hatt hovedansvaret for backend serveren som vi har satt opp gjennom RabbitMQ [3.1.6]. Serveren håndterer events som kommer fra markedsplassen til Fixrate.

Det ble ikke tildelt noen faste roller gjennom bacheloroppgaven, men heller et tett samarbeid og god kommunikasjon som har sørget for lik arbeidsfordeling slik at de forskjellige oppgavene ble utført. En rollefordeling ville vært mer aktuelt hvis teamet hadde bestått av flere medlemmer.

3.3 Utviklingsprosess

Før oppstart av bacheloroppgaven ble det diskutert ulike smidige arbeidsmetodikker som vi kunne ta i bruk under utviklingen av applikasjonen.

3.3.1 Scrum

Begge har tidligere jobbet med den smidige metoden scrum og likte måten prosjekter ble gjennomført på ved bruk av scrum. Scrum er en metodikk ment for team med størrelse på 4-9 personer med en rekke aktiviteter både før, under og etter hver sprint, og er derfor lite hensiktsmessig for små team. Arbeidsmetodikken i scrum består av noen viktige punkter:

- Splitte arbeidet til små oppgaver å sortere de etter prioritet og estimert arbeidstid.
- Lage burndown-chart for å vektlegge arbeidsoppgaver
- Sette en fast start- og stopptid på hver sprint
- Etter hver sprint skal kode være klar for levering og demonstrasjon
- Ha en gjennomgang i teamet etter endt sprint om hva som kan optimaliserer for å oppnå bedre resultater neste sprint ved å fortelle hva som gikk bra/dårlig under sprinten.

3.3.2 Kanban

Kanban ble også diskutert og har sin karakteristikk i tavler med god oversikt over arbeidsoppgaver. Tavlen er delt inn i kolonner med hva som skal gjøres, pågående og ferdigstilte oppgaver. Kanban skiller seg fra scrum på flere områder hvor en av de er tilnærminger til endringer. I scrum skal endringer gjøres etter en sprint, men i kanban begynner man der man er og utvikler litt om gangen. På grunn av dette blir arbeidsflyten forskjellige fra scrum til kanban. I scrum velger man ut alle arbeidsoppgaver som skal fullføres gjennom en sprint. Deretter setter man en sluttdato for når alle oppgavene skal være fullførte i motsetning til kanban som ikke består av sprinter. Kanban sine viktige punkter i arbeidsmetodikken består av å:

- Splitte arbeidet i mindre oppgaver
- Skrive hver oppgave på ett kort å feste på tavlen
- Bruke kolonner på hver tavle for å illustrere hvor i prosessen oppgaven befinner seg.

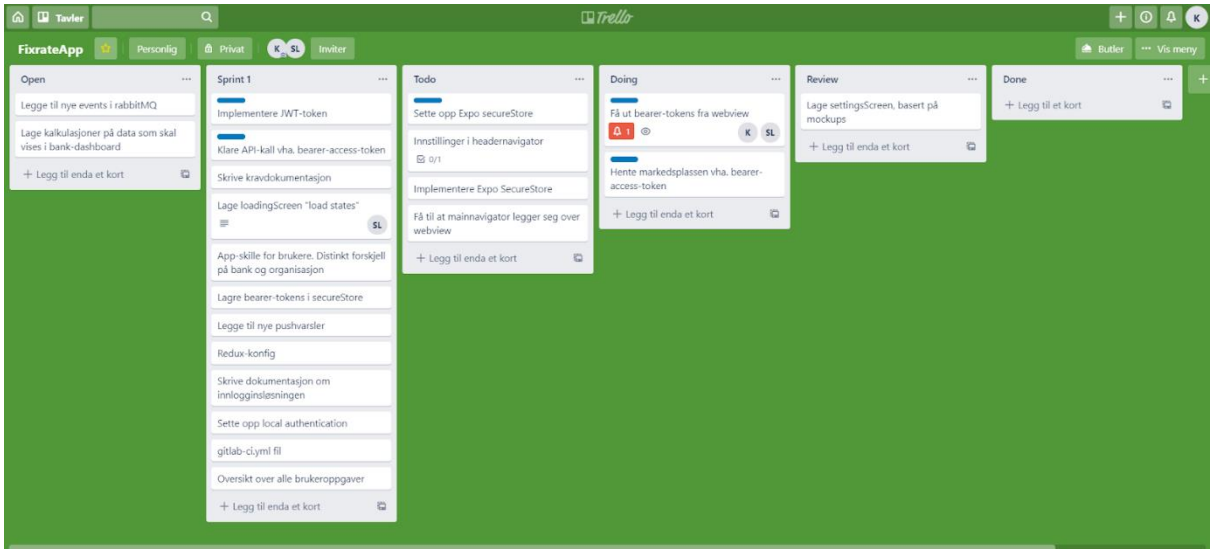
3.3.3 Scrumban

Scrumban kombinerer strukturen til scrum med arbeidsflyten som kanban tilbyr. Noen nøkkelpunkter som brukes i scrumban er:

- Bestemme hvor mye arbeid som er reelt å fullføre innenfor sprinten.
- Prioritering av oppgaver
- Individuelle roller er ikke spesifisert
- Bruk av tavler for å holde kontroll på arbeidsprosessen og oppgaver

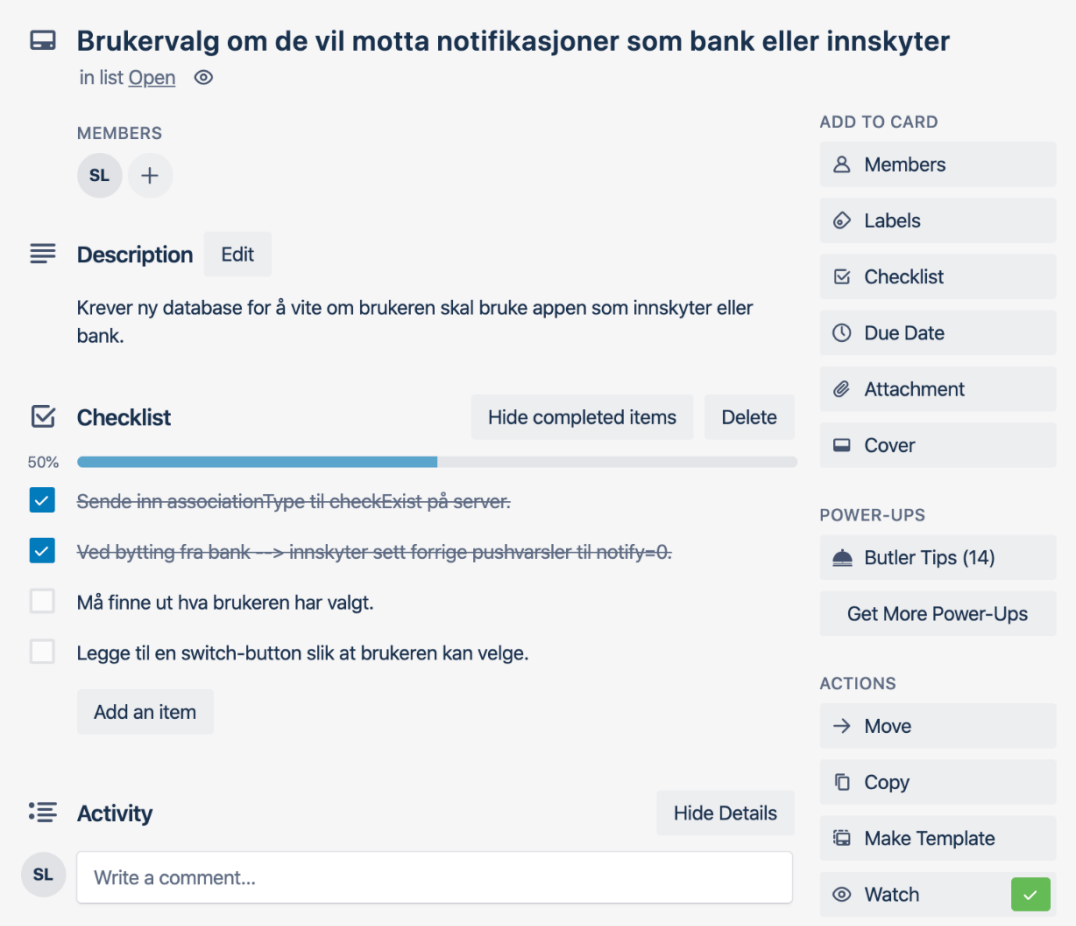
Etter litt diskusjon med veileder ble vi enige om å bruke scrumban som arbeidsmetodikk på det grunnlag av at vi kun var to på teamet, det var rom for å tilføye endringer til krav av systemet underveis og vi var ikke nødt til å bruke timeestimering. Dette var veldig viktig for oss siden vi begge hadde fag ved siden av bacheloroppgaven etterfulgt av en eksamen midt i semesteret.

Vi delte utviklingen av applikasjonen inn i fem segmenter. Segmentene bestod av innlogging, pushvarsler, visning av innhold,gjøremålslisten og dashboard. Hvert segment ble sett på som sin egen sprint med egen start- og stopptid. Hvert segment fikk deretter use-cases tilhørende det området av applikasjonen de hørte hjemme i. Siden en del av Scrumban er å ha en tavle hvor oppgaver føres opp ble alle oppgaver lagt inn i den virtuelle tavle-løsningen Trello. Nedenfor er et skjermbilde av tavlen vår under sprint 1. Vi har lagt til en egen kolonne "review" slik at vi enkelt kan kvalitetssikre hverandres arbeid før vi anser det som ferdigstilt.



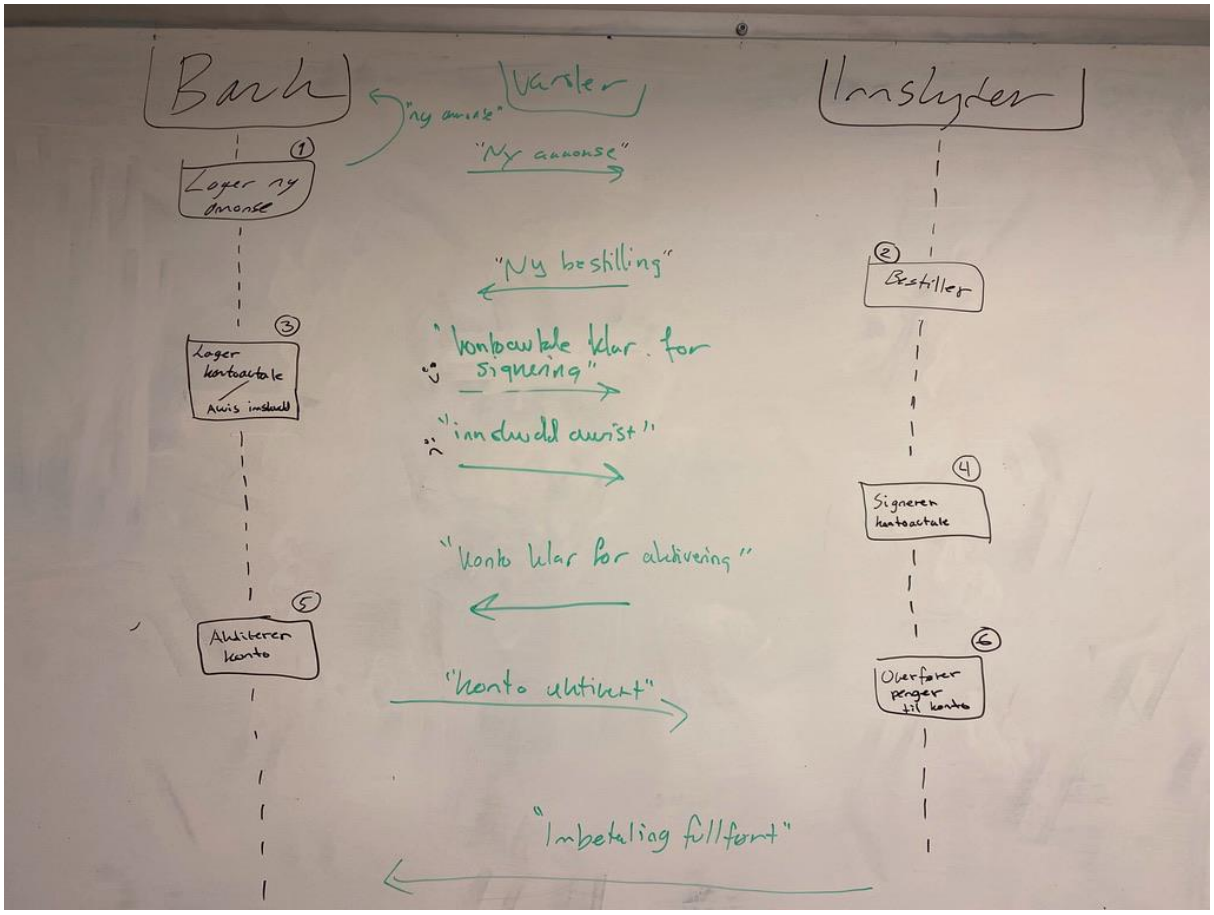
Figur 3: Skjermbilde fra Trello av Scrumban-board i sprint 1 av utviklingen

Tavlen består av flere oppgaver som er presentert som kort med en tilhørende arbeidstittel. Hvert kort kan også inneholde mindre oppgaver som må utføres for å få fullført oppgaven. Nedenfor er detaljer om et kort og de mindre oppgavene som må gjennomføres.

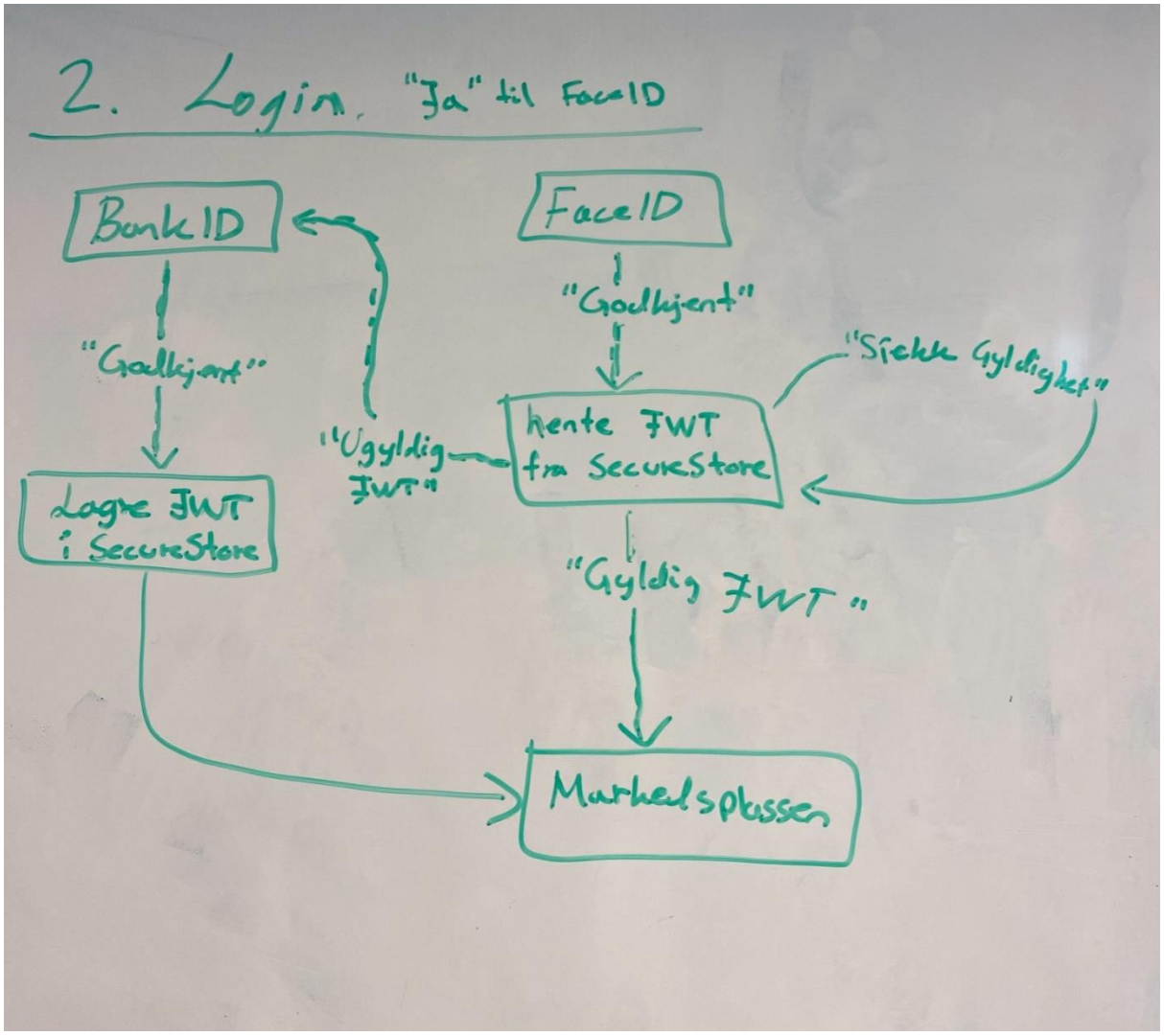


Figur 4: Skjermbilde av hvordan et kort kan ha flere mindre oppgaver

Videre under utviklingen hadde vi til dels god tilgang til whiteboard på kontorene til Fixrate. Disse ble godt brukt under planleggingen av sprinter for å få en god oversikt over hvilke utfordringer sprinten inneholdt. Ved oppstart av sprinter valgte vi derfor å bruke whiteboards for kartleggingen av arbeidet og hva som måtte gjennomføres. Nedenfor er to bilder av whiteboardet etter to sesjoner hvor vi har kartlagt hvilke pushvarsler vi trenger gjennom bestillingsprosessen i applikasjonen og en use-case som vi fant når man skal logge inn i applikasjonen.



Figur 5: Bilde av whiteboard under kartlegging av pushvarsler

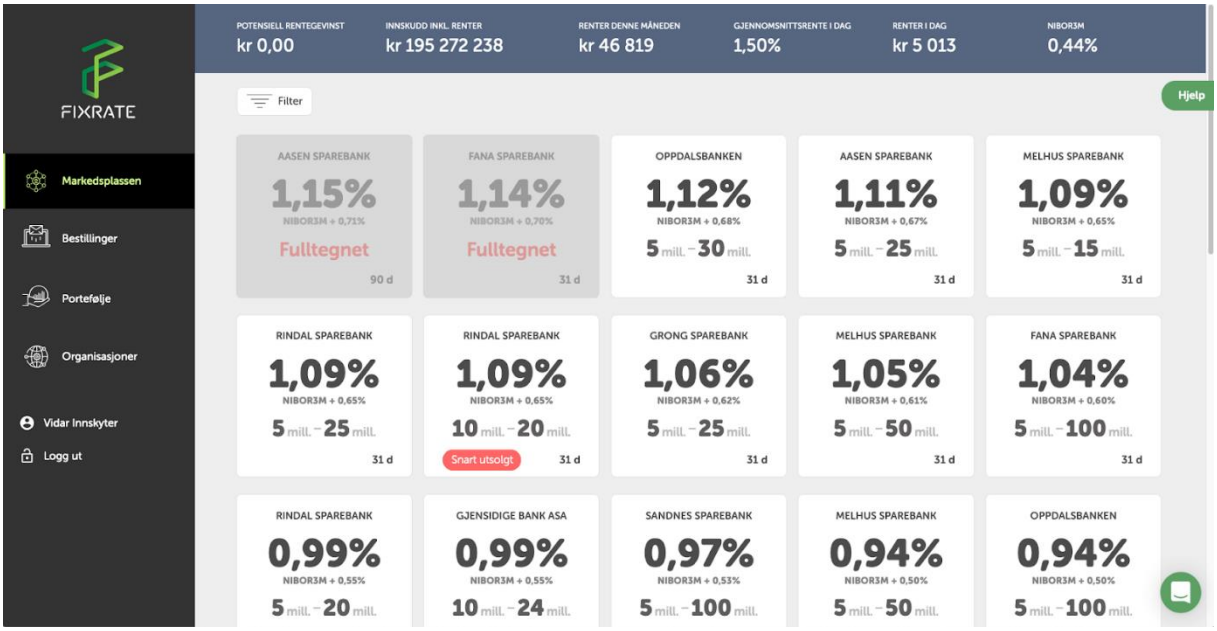


Figur 6: Bilde av en use-case i innloggingen tegnet opp på et whiteboard

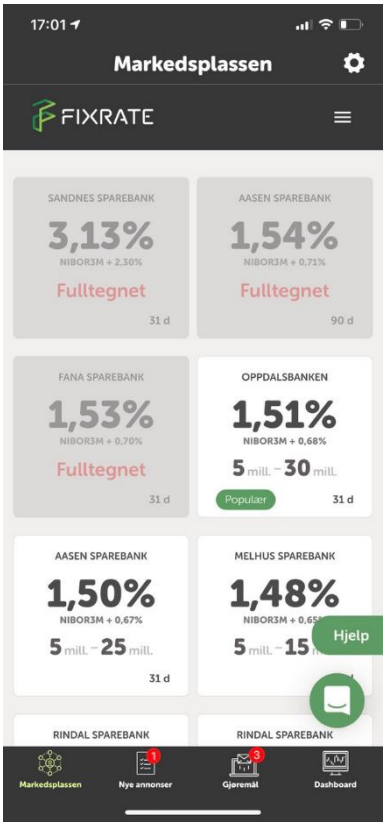
3.4 Design

Under utviklingen av applikasjonen var design og brukeropplevelse et viktig tema. Fixrate hadde selv et ønske om å vise innhold fra nettsiden direkte i applikasjonen. Dette var ønskelig fordi ved endringer på design av sentralt innhold på nettstedet ville det vises i applikasjonen uten å måtte endre kildekoden til applikasjonen. Siden applikasjonen er ment som et tilskudd til det eksisterende nettstedet med mye likt innhold og funksjoner er det derfor viktig å ikke introdusere nye elementer til kundene. Ved å ha et konsistent design mellom nettstedet og applikasjonen vil det bli intuitivt og enkelt for brukerne å ta den i bruk uten å måtte bli vant til nye elementer.

Ved å ta i bruk samme fonter, farger og elementer som brukeren er vant til å både se og agere på i applikasjonen kan vi oppnå et konsistent design på tvers av plattformene. I applikasjonen har vi derfor tatt i bruk samme fonter og replikert elementer fra nettstedet for å oppnå dette. Nedenfor er det to skjermbilder som viser markedsplassen til Fixrate på nettstedet og i applikasjonen.



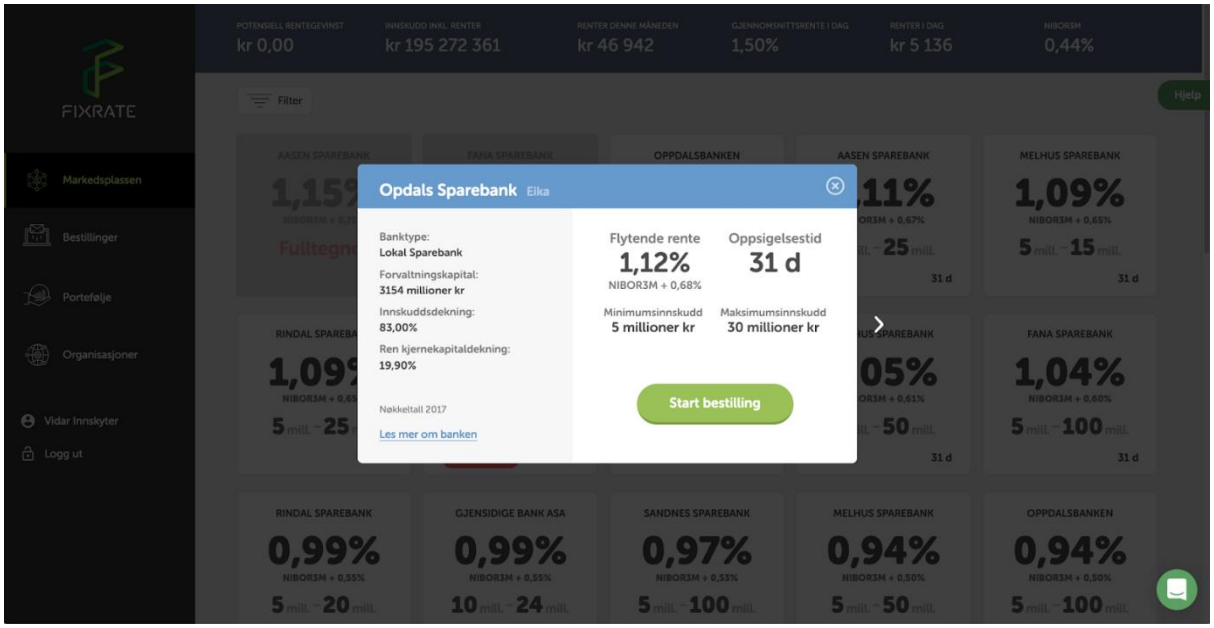
Figur 7: Skjermbilde av markedsplassen på nettstedet til Fixrate



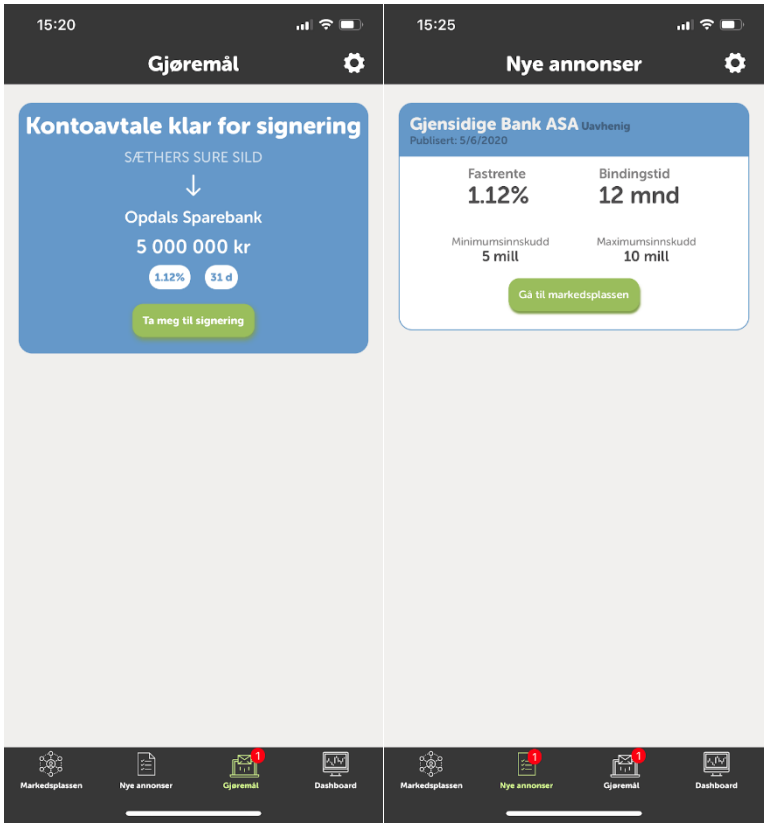
Figur 8: Skjermbilde av markedsplassen i applikasjonen

Som nevnt over ønsket Fixrate at vi skulle vise noe innhold direkte fra nettsiden deres. Dette kommer frem på markedsplassen der vi viser den mobile versjonen av nettstedet gjennom et WebView som kan vise hvilken som helst nettside ved å spesifisere en URL.

Siden markedsplassen er preget av et design som presenterer hver annonse i form av kort ønsket vi å ta dette videre med oss i designet vårt. I applikasjonen har vi egne sider som viser nye annonser og brukeren sine gjøremål. Her har vi replikert mye av det eksisterende designet fra nettstedet. Nedenfor er det tre skjermbilder som viser designet til hvordan en annonse presenteres når du trykker på den i nettstedet til Fixrate, designet til et nytt gjøremål og ny annonse i applikasjonen.



Figur 9: Skjermbilde av detaljer til en ny annonse på nettstedet til Fixrate



Figur 10: Skjermbilde av ny annonse i applikasjonen

Figur 11: Skjermbilde av nytt gjøremål i applikasjonen

Som vist i skjermbildene over har vi prøvd å oppnå et konsistent og sømløst design mellom nettsted og applikasjon ved å bruke kjente elementer, tekstfonter og farger som brukeren allerede er vant til å se på det eksisterende nettstedet.

4 Resultater

Dette kapittelet tar for seg de resultatene som vi har kommet fram til under utførelsen av bacheloroppgaven. Kapittelet er delt inn i tre underkapitler, vitenskapelige resultater, ingeniørfaglige resultater og administrative resultater.

4.1 Vitenskapelige resultater

4.1.1 Produkt

Applikasjonen er ment som et tilskudd til det eksisterende nettstedet til Fixrate. Det har vært fokus på at innholdet i applikasjonen ikke skal erstatte noe av den funksjonaliteten som allerede eksisterer i Fixrate sitt system og heller ikke introdusere nye elementer for å forvirre brukeren. Vi deler innholdet inn i underkapitler basert på de største implementasjonene som er gjort i applikasjonen.

Innlogging med BankID og biometri

Tidligere i kapittel 1.1 ble det nevnt at innloggingen var ufullstendig ved levering av høstprosjektet og det ble derfor vår første prioritet ved oppstart.

Den endelige løsningen som vi implementerte i applikasjonen ved levering bruker den samme innloggingssiden som brukes på nettstedet til Fixrate. Vi kom fram til en løsning der vi ikke trengte å lage et tredjeparts system som tok for seg kryptering og utveksling av nøkkelpar mellom bruker og system.

BankID

Løsningen i applikasjonen tar i bruk nesten samme innloggingsmetode som nevnt i kapittel 2.5 om BankID. Det er derimot noen endringer. Hvis man ser på stegene for SAML-innloggingen så griper applikasjonen inn mellom steg 4 og 5. Her blir SAML-responsen fra Signicat fanget opp av applikasjonen og sendes deretter til et nytt endepunkt gitt av Fixrate. SAML-responsen blir deretter verifisert av serveren til Fixrate og sender tilbake en JWT som kan brukes videre for å vise innhold fra Fixrate sitt nettsted som krever autentisering.

Mer detaljert informasjon om hvordan løsningen angående bruken av JWT kommer i neste underkapittel om Biometrisk innlogging siden det er direkte knyttet mot løsningen til biometrisk innlogging.

Biometrisk innlogging

Den biometriske innloggingen var nødt til å ta i bruk deler av den eksisterende BankID-innloggingen som beskrevet over. For at brukeren skulle kunne bruke funksjoner, se innhold og motta riktig data var vi derfor nødt til å ha en vedvarende gyldig sesjon i applikasjonen. For å oppnå en slik løsning var vi derfor nødt til å finne løsninger for hvordan vi kunne oppnå nettopp dette. Flere alternativer ble diskutert mellom oss og utviklingsteamet til Fixrate. Det ble avholdt en liten workshop for å komme frem til en løsning, der et alternativ var å bruke nøkkel-par for utdeling av autorisering ble diskutert og undersøkt.

I workshopen ble PKCE, også kalt PIXY, som står for Proof Key for Code Exchange sett en del på. For en detaljert oversikt over stegene i denne metoden for utveksling av autorisasjonsdata se vedlagt dokument om innlogging [33]. I løpet av workshopen ble det foreslått at Fixrate heller kunne utlevere en JWT som inneholder en gyldig sesjon ved å sende SAML-responsen til et nytt og eget endepunkt kun ment for dette formålet. Sistnevnte ble brukt i den endelige implementasjonen for å oppnå biometrisk innlogging i applikasjonen.

Etter at applikasjonen mottar denne JWTen må den lagres sikkert og lokalt på mobiltelefonen til brukeren. Det ble tatt i bruk et bibliotek, SecureStore, som lagrer nøkkelpar med navn og tilhørende verdi trygt i mobiltelefonens eget område for lagring av sensitiv data. Operativsystemet til mobiltelefon bestemmer hvordan dette lagres og krypteres og er noe forskjellig på Android og iOS. Dette området er deretter låst og beskyttet av et annet bibliotek, LocalAuthentication, som håndterer den biometriske innloggingen. Biblioteket tilbyr funksjoner som kan sjekke om telefonen har gyldig maskinvare for biometri og for å ta denne maskinvaren i bruk. Maskinvaren for bruk av FaceID kan ikke brukes under utviklingen siden det krever at applikasjonen blir bygd som en frittstående app. For å få tilgang til JWTen i applikasjonen er brukeren nødt til å autentisere seg med enten passord, fingeravtrykk eller ansiktsgjenkjenning. Se Systemdokumentasjon [30] for ytterligere informasjon om bibliotekene SecureStore og LocalAuthentication.

Som nevnt tidligere i underkapittelet om BankID bruker Fixrate i dagens eksisterende løsning en iFrame for innloggingen. React-Native har en lignende komponent som heter WebView. WebView gjør akkurat det samme som en iFrame, men er ment for mobiltelefoner. WebView tilbyr også en rekke viktige funksjoner som vi har tatt i bruk. Vi kan injisere vårt eget JavaScript i nettsiden som vises i WebViewet og stoppe den normale oppførselen etter at brukeren har logget inn. Nedenfor er et skjermbilde av koden vi injiserer i vårt WebView.

```
const injectedJavaScript = `
      let jwt = "";
      if (loginStatus) {
          let xhr = new XMLHttpRequest();

          xhr.open("GET",'${JWTurl}');
          xhr.responseType = 'text/plain';
          xhr.send();
          xhr.onload = function() {
              if (xhr.status === 200) {
                  let response = xhr.response;
                  jwt = xhr.response;
                  window.ReactNativeWebView.postMessage(jwt, '*');
              }
          }
      }
      true`;
```

Figur 12: Skjermbilde av injisert JavaScript i WebView

Ved en vellykket innlogging vil WebViewet få en respons om innloggingen var vellykket. Deretter mottar WebViewet en SAML-response fra Signicat som nevnt i punkt 5 i underkapittelet om BankID. Den injiserte koden oppretter deretter en XMLHttpRequest mot et nytt endepunkt Fixrate har laget. SAML-responsen sendes inn til endepunktet, blir verifisert og sender deretter en JWT tilbake til applikasjonen. Brukeren får deretter spørsmål om han/hun vil bruke biometrisk innlogging neste gang. Velger brukeren ja blir JWTen lagret i SecureStore slik at den kan hentes ut ved hver innlogging.

Neste gang brukeren logger inn vha. biometri vil applikasjonen åpne SecureStore, hente ut JWTen og sende inn til Fixrate for å verifisere gyldigheten. JWTen gjør det derfor mulig å vise hvilket som helst innhold fra Fixrate sitt nettsted som trenger autorisering gjennom et WebView i applikasjonen, og man unngår BankID-innlogging hver gang applikasjonen skal brukes. Ved å bruke biometri oppnår man også en større brukervennlighet da det oppfattes som tungvint på mobil å autentisere seg ved BankID hver gang man skal bruke en applikasjon. For å oppnå en tryggere sikkerhet i applikasjonen etter innloggingen er det også lagt inn funksjonalitet som krever re-autentisering ved inaktivitet i over femten minutter.

Innstillinger

Det er lagt til et tannhjul øverst i høyre hjørne hvor brukeren enkelt kan nå siden som viser innstillingene for brukeren. I innstillingene kan brukeren velge hvilke deler han vil motta pushvarsler på. Dette er gjort på lik måte som på nettstedet, men i stedet for at brukeren kan

trykke av eller på for varsling på epost, er det pushvarsler brukeren kan velge mellom i applikasjonen. Hvert pushvarsel er representert i form av en switch-knapp [figur 16] hvor brukeren kan velge om de vil ha valgt type varsel aktivert eller ikke. Innstillingene er koblet til hver enkelt bruker i en database og endres når brukeren trykker på dem. Varslingene på e-post og pushvarsler er ikke knyttet sammen, da de ikke har noe med hverandre å gjøre, men er likevel like i design slik at brukeren kan forvente samme oppførsel på push-innstillinger som på nettstedets innstillinger for e-post.

På innstillinger kan brukeren også velge å aktivere eller deaktivere innlogging med biometri. Dette har vi implementert som et valg brukeren kan ta for at brukeren ikke skal bli låst mot en løsning. Innstillinger inneholder også en knapp slik at brukeren kan logge ut av applikasjonen.

Markedsplassen

Det første brukeren møter når de logger inn i applikasjonen er markedsplassen. Den vises gjennom et WebView som vi bruker for å vise den samme markedsplassen som på nettstedet. Markedsplassen er, som nevnt tidligere, et marked for store bankinnskudd. Brukeren kan trykke på en annonse for å se detaljer om tilbudet. Deretter kan brukeren velge å starte en bestilling om det er ønskelig. Den dataen som sendes ved agering på en annonse i markedsplassen, vil da sendes både til systemet til Fixrate, og det systemet som vi har laget. Uavhengig av hvor du som bruker velger å agere på en annonse - om det er gjennom applikasjonen eller nettstedet - så vil dataen som lages, trigges, eller sendes, gå over begge systemene.

Nye annonser

I applikasjonen har vi laget en egen dedikert side for visning av alle nye annonser fra de siste syv dagene. På nettstedet kan annonsene få merkelapper som beskriver om en annonse er “ny”, “populær”, eller “snart utsolgt”. Annonsen merkes som “ny” med en gang den legges ut og “populær” så snart det blir gjort bestillinger. Videre blir annonsen merket som “snart utsolgt” når annonsens totale etterspørsel begynner å nærme seg. I applikasjonen har vi definert nye annonser som alle annonser som er maksimalt syv dager gamle. Dette er for å gi brukeren en oversikt over de nyeste annonsene som ligger ute på markedsplassen siden “ny”-labelen forsvinner så snart noen bestiller.

Gjøremålsliste

På samme måte som med nye annonser har vi laget en egen dedikert side for visning av gjøremålene til en bruker. Her vil det dukke opp gjøremål som er basert på stegene som følger bestillingsprosessen [1.1.2] av et innskudd hos bank. Et gjøremål i gjøremålslisten har en beskrivende tittel og kan for eksempel være “Kontoavtale klar for signering”. Dette gjøremålet er da tilegnet en innskyter hos en organisasjon. Kortet vil ha utfyllende informasjon om gjøremålet som hvilken organisasjon det gjelder for og hvilken bestilling dette gjelder. På hvert gjøremål er det opplyst hvem fra organisasjonen som har rettigheter til å for eksempel signere kontoavtalen. Hvis en bruker har rettigheter for å signere avtalen kan han/hun trykke på en knapp på gjøremålet, og blir deretter tatt inn på en egen side for signering gjennom et WebView.

Dashbord

I toppen på nettstedet er det en liste som viser en overordnet oversikt av organisasjonens eller bankens totale gevinst gjennom alle sine innskudd. Den inneholder totale innskuddsbeløp inklusive renter, hvor mye som er opptjent gjennom renter denne måneden, og den viser gjeldende NIBOR3M. Denne listen har det vært stor interesse for hos kundene til Fixrate og er noe de er veldig opptatte av. Denne har vi også implementert i applikasjonen. På toppen av siden i applikasjonen ble det lagt til et vindu der brukeren kan swipe til høyre eller venstre for å se de forskjellige delene av dashbordet i applikasjonen. All dashboard-data hentes ut fra Fixrate sin server ved innlogging. Deretter kjøres det kalkulasjoner på disse for å få en stigning i tallene slik som på nettstedet. Grunnen for at det ikke ble utviklet en dedikert widget for iOS og Android er at det i høsten 2020 skal komme en stor oppdatering for iOS med et større fokus på widgets [31]. Det er stor grunn til å tro at man må gjøre store endringer på utviklingen rundt en eventuell widget når denne oppdateringen kommer og det ble derfor satt til siden. Ingenting er bekreftet av Apple [40] enda, men det ventes nærmere informasjon rundt dette i juni.

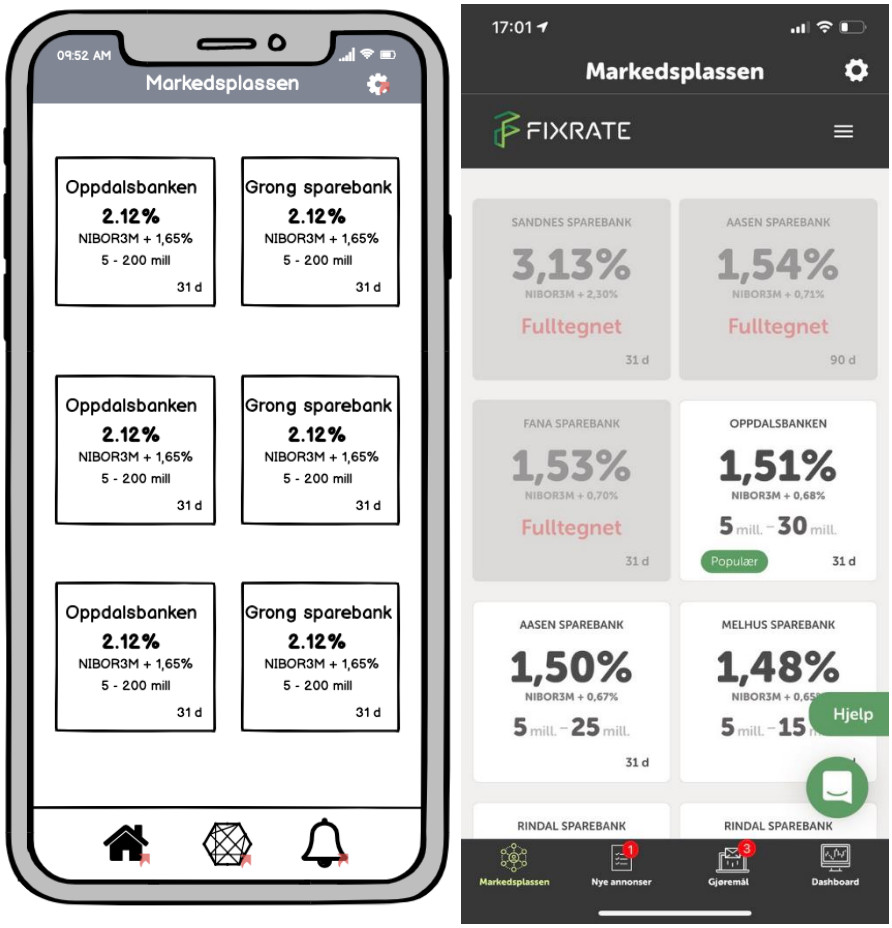
Pushvarsler

Får å få sendt pushvarsler til brukerne er vi nødt til å identifisere hver enkelt brukers telefon. Når brukeren logger inn, lagrer vi telefonens unike id. Identifikasjonen til telefonen kalles i systemet for expo-token. Det er disse vi må bruke når vi skal sende ut pushvarsler til brukerne. Videre har vi koblet oss på RabbitMQ serveren til Fixrate. Resultatet av dette er at vi får tak i de eventene som aktiveres når en kunde bruker nettstedet. Etter oppkoblingen mot RabbitMQ serveren til Fixrate, er vi nødt til å identifisere de eventene som er relevante for den riktige type hendelse som ble utført på nettstedet. Eventene som kommer fra nettstedet inneholder forskjellige ider og en variabel som forteller oss hva slags type hendelse det er snakk om. Med id vi får fra eventene kan vi hente ut den dataen vi trenger for å fylle ut pushvarslene med informasjon angående hendelsen. Den nødvendige dataen vi trenger hentes deretter ut fra databasen til Fixrate. Siden applikasjonen skal brukes som et tillegg til nettstedet er det nødvendig at vi henter ut den samme dataen som brukes på nettstedet. Vi fikk også tilgang til Fixrates SMTP-server som viser innholdet til e-postene som sendes ut til kundene slik at vi kan se hva slags innhold som ble sendt på e-post. Slik kan vi replikere pushvarslene til å inneholde det samme innholdet som e-postene og bedre forstå hva som er relevant å sende ut til brukerne. Videre sender vår RabbitMQ server et REST-kall mot Expo sitt API for sending av pushvarsler. Når pushvarsler sendes ut henter RabbitMQ serveren vår ut den nødvendige dataen om hendelsen fra Fixrates database, alle unike expo-tokens for de aktuelle brukerne som abonnerer på hendelsen, og lager deretter et json-objekt av dette. Deretter sendes objektet til et endepunkt gitt av expo som håndterer utsendelse av pushvarsler og brukerne mottar deretter et pushvarsel på sin telefon.

4.1.2 Design

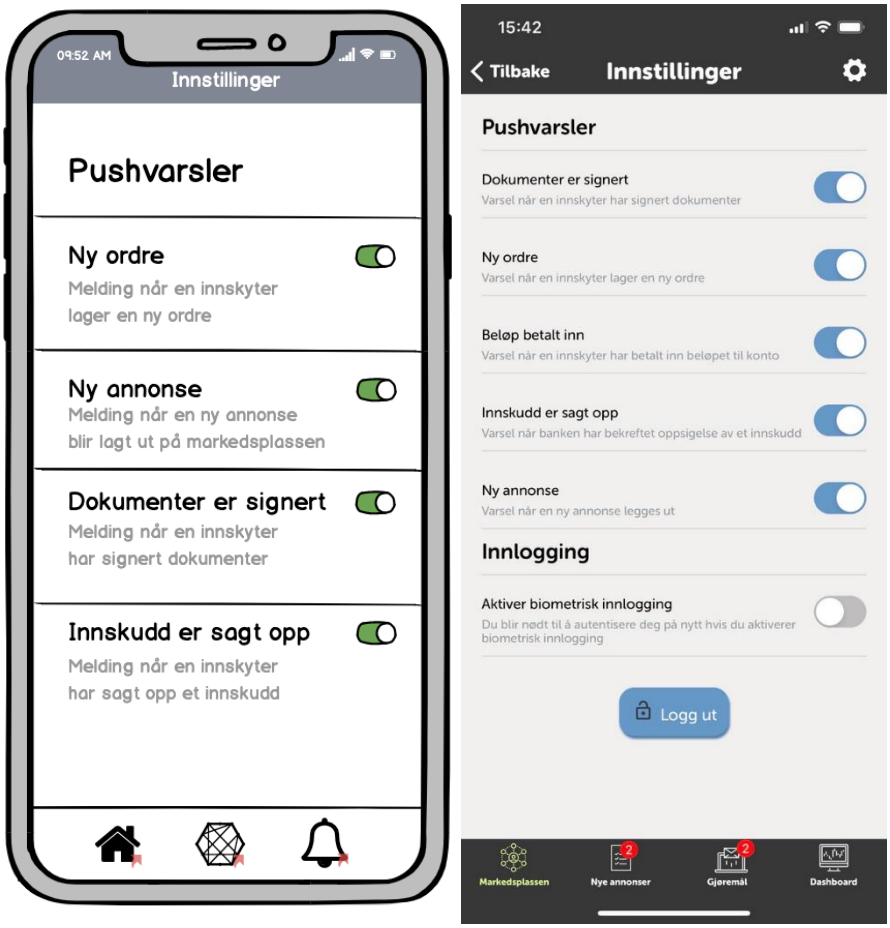
Ved starten av bacheloroppgaven ble noe av det første vi jobbet med mockups for applikasjonen. Disse skulle lages gjennom flere iterasjoner, der hver iterasjon ble avsluttet med å vise disse frem til oppgavestiller. I dette underkapittelet skal vi se på hvordan designet var tenkt ved starten gjennom wireframes og til faktisk design ved levering av oppgaven. Skjermbildene over markedsplassen er lik både hos bank og innskyter og vil derfor kun bli vist en gang. Siden applikasjonen vår består av to typer brukere, bank og innskyter, har vi derfor valgt å kun vise skjermbilder av wireframes fra siste iterasjon. For å se alle iterasjoner se Wireframes [39] som inneholder alle filene lagd i Balsamiq.

Bank



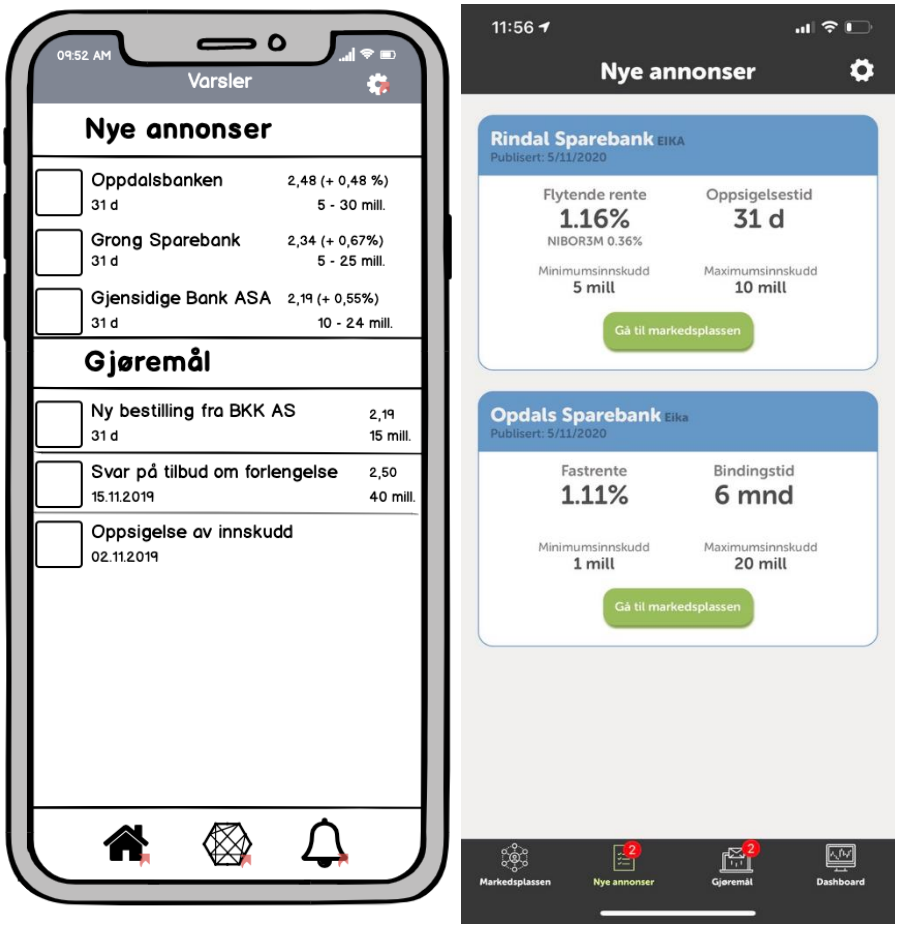
Figur 13: Skjermbilde av markedsplassen ved levering for bank

Figur 14: Wireframe fra tredje iterasjon av markedsplassen for bank



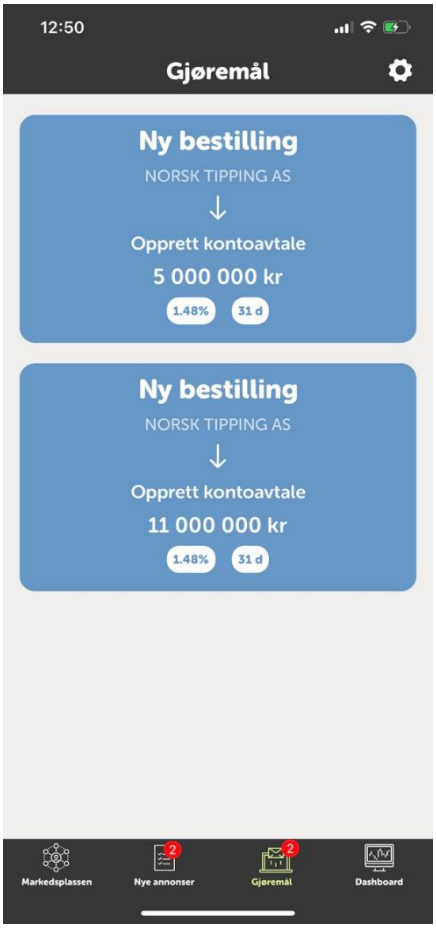
Figur 15: Skjermbilde av siden for innstillinger for bank ved levering

Figur 16: Wireframe fra tredje iterasjon av meny for innstillinger for bank



Figur 17: Skjermbilde av siden til nye annonser ved levering for både innskyter og bank

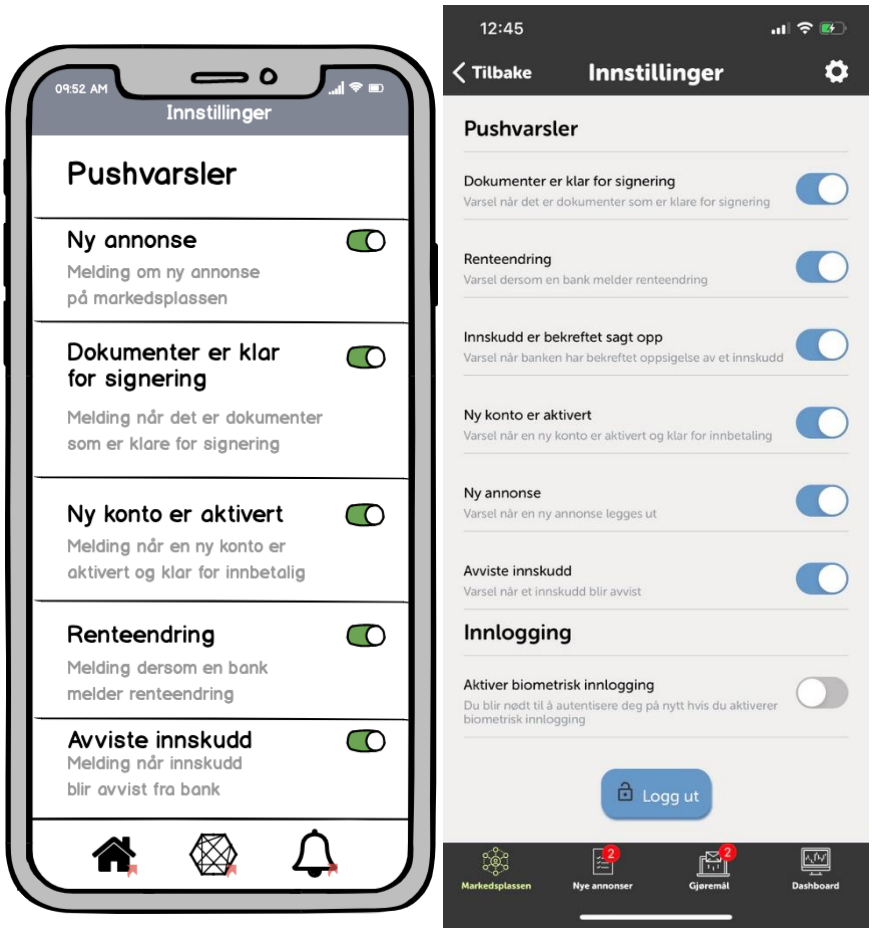
Figur 18: Wireframe fra tredje iterasjon av listen for nye annonser og gjøremål



Figur 19: Skjermbilde av siden over gjøremål for bank ved levering

Figur 17, 18 og 19 kan sees på i sammenheng. Mot slutten av bacheloroppgaven hadde vi hyppigere møter med oppgavestiller for å få tilbakemeldinger på designet til applikasjonen. Det ble under utviklingen bestemt at brukernes gjøremål og nye annonser skulle skilles ut i sine egne respektive sider for bedre oversikt. Dette var fordi at en ny annonse i våre øyne, ikke kunne anses som en brukers gjøremål.

Innskyter

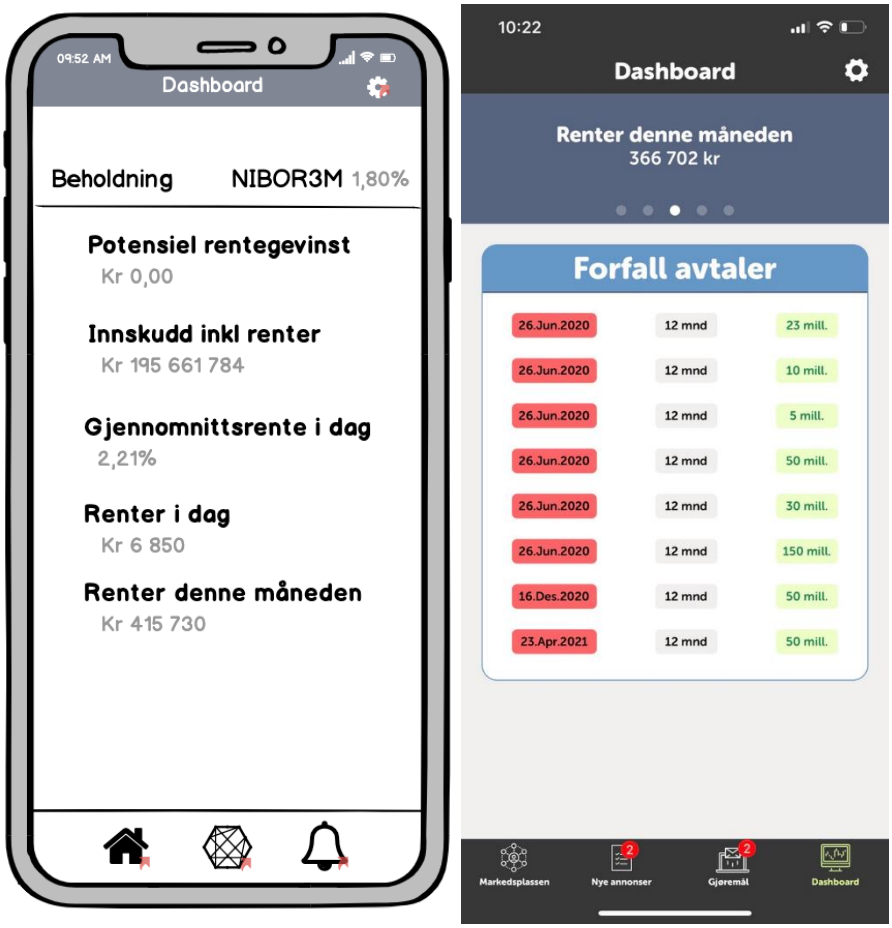


Figur 20: Skjermbilde av siden for innstillinger til innskyter ved levering

Figur 21: Wireframe siden for innstillinger til innskyter

Figur 22: Skjermbilde av siden til gjøremål for innskyter ved levering

Figur 23: Wireframe fra tredje iterasjon av listen for nye annonser og gjøremål



Figur 24: Skjermbilde av dashboardet til innskyter ved levering

Figur 25: Wireframe fra tredje iterasjon av dashboardet til innskyter

Videre av skjermbildene kan man også se at det kommer klart frem hvor i applikasjonen brukeren befinner seg ved å se på headeren i figur 25 hvor det står "Dashboard". Det samme kommer fram i navigasjonsbaren nederst på skjermen. I navigasjonsbaren blir gjeldende side markert ved at fargen til ikonet endrer seg fra hvit til grønn. Ved å bytte fargene på ikoner ved navigering mellom de forskjellige sidene skaper vi en tilbakemelding og signaliserer at brukeren har skiftet side.

4.2 Ingeniørfaglige resultater

I dette kapittelet tar vi for oss de målene som ble satt i starten av bacheloroppgaven, målene er hentet fra visjonsdokumentet [1]. Vi beskriver status for hver av dem ved levering av bacheloroppgaven. Her har vi tatt med de målene som er mest aktuelle og relevante for produktet og dagens status.

4.2.1 Mål

1. ***Gjennomføre bestilling på markedsplassen***

Den foreslåtte løsningen var å implementere en dedikert side for å gjennomføre en bestilling etter brukeren har valgt en annonse på markedsplassen.

Løsningen skulle bestå av en egen side i applikasjonen hvor brukeren kunne gjennomføre en bestilling av innskudd på en annonse. Etter noen iterasjoner med oppgavestiller og utviklerteamet ble det diskutert hvordan Fixrate kunne vedlikeholde applikasjonen i fremtiden. Her ble det sett på løsninger slik at Fixrate skulle slippe å gjøre endringer i kildekoden til både applikasjon og nettsted. Det resulterte i at vi endret på hvordan brukeren kunne gjennomføre en bestilling ved å bruke et WebView som viser nettstedets markedsplass. Gjennom dette WebViewet vil brukeren få muligheten til å starte en bestilling på annonse, på lik måte som det er på nettstedet. Målet med at brukeren skal kunne gjennomføre bestilling på markedsplassen er i seg selv oppfylt, men det ble løst på en annen måte enn det som vi hadde definert og planlagt at skulle løses fra start av bacheloroppgaven.

2. ***Innlogging vha. biometri***

Dette målet ble ansett som både viktig og kritisk. Fra tidligere arbeid fungerte ikke innloggingen på telefoner med iOS og det ble derfor vår første prioritet ved oppstart av bacheloroppgaven. Målet er at brukeren skal slippe å måtte autentisere seg med BankID hver gang han skal logge seg inn i applikasjonen. Systemet tillater brukeren å ta i bruk biometrisk maskinvare som er tilgjengelig på telefon for fremtidige innlogginger etter førstegangs autentisering med BankID. Systemet sjekker hvilken type biometrisk hardware som er tilgjengelig på brukerens telefon og lar deretter brukeren velge om de ønsker å bruke denne løsningen. På de nyeste smarttelefoner er det vanlig med biometrisk maskinvare som enten er Face- eller TouchID.

3. ***Samle alle gjøremål i en egen side for enklere oversikt***

Systemet har implementert en egen side i applikasjonen som inneholder en oversikt over gjøremålene. Om brukeren ikke har noen gjøremål vil denne siden være tom og erstattet med en tekst som forteller brukeren at det er ingen aktive gjøremål. Etterhvert som det startes en prosess ved bestilling av innskudd, eller at banken sender en renteendring på et aktivt innskudd, vil et tilhørende gjøremålet dukke opp på denne siden. Som ekstra funksjonalitet er det implementert en pull-to-refresh som oppdaterer siden med gjøremål.

4. ***Liste over alle varsler som er mottatt***

Løsningen inneholder ikke en dedikert side som viser alle mottatte pushvarsler hos brukeren. Siden det varsles om enten en ny annonse eller et nytt gjøremål har de hver sin respektive side i applikasjonen. I stedet for en oversikt over nye varsler er det et ikon på navigasjonsbaren som viser hvor mange nye annonser/gjøremål som er i listen.

5. ***Tildele gjøremål/oppgaver til andre brukere, Tildele oppgaver til seg selv***

I planleggingsfasen diskuterte vi en løsning hvor brukere med de rette rettighetene kunne tildele oppgaver til andre eller seg selv, men etter samtale med både veileder og oppgavestiller, ble vi enige om at dette kunne være en "nice to have" funksjonalitet som vi går bort fra. Utviklerteamet til Fixrate mente også at en slik funksjon også ville være veldig funksjonelt krevende å implementere. Systemet i dag viser hvem av brukerne som har rettigheter til å agere på de forskjellige gjøremålene.

6. ***Motta pushvarsler når brukeren har blitt tildelt et gjøremål/oppgave***

Som nevnt i punkt 5 ble ikke denne funksjonaliteten implementert. Systemet har derfor ingen løsning hvor brukeren mottar pushvarsel ved tildelt gjøremål/oppgave. Dagens løsning sender ut pushvarsel til alle brukere av en organisasjon når det dukker opp et nytt gjøremål for organisasjonen.

7. ***Motta pushvarsler når en ny bestilling mottas***

Hensikten med dette målet er å oppdatere bank-brukerne av systemet. Når en innskyter har bestilt et innskudd gjennom en av bankens annonser på markedsplassen, vil systemet sende ut et pushvarsel. Pushvarselen blir sendt ut til alle brukere innad banken som mottok bestillingen og vil deretter legges inn i brukerens gjøremålsliste inntil gjøremålet er behandlet.

8. ***Overordnet oversikt over egen portefølje***

I planleggingsfasen var det ønsket å få til en mobiltilpasset portefølje som en egen dedikert side. Målet her var ikke satt som prioritet, men heller en funksjonalitet som hadde vært fint og hatt med i applikasjonen. Systemet i dag har ikke implementert en egen dedikert side for visning av portefølje, da porteføljen som vises på nettstedet er under utvikling og oppdateres stadig med nytt design og ny funksjonalitet. Etter avtale med oppgavestiller har vi valgt å se bort fra denne implementasjonen.

9. ***Motta pushvarsel ved renteendring på aktive innskudd***

Pushvarsel ved renteendring på aktive innskudd er implementert i systemet. Alle brukere fra en organisasjon med et aktivt innskudd liggende hos bank vil motta pushvarsler når banken melder renteendring. Systemet vil deretter legge renteendringen til i gjøremålslisten til brukerne.

10. Signere kontoavtale

Det har vært mye diskutert om brukeren skal få mulighet til å agere på gjøremål i applikasjonen. Systemet har ved levering det slik at brukeren kan gjøre dette på noen steg i bestillingsprosessen. Eksempelvis signering av en kontoavtale. Brukeren blir tatt videre inn i et WebView som viser den siden på nettstedet som lar brukeren signere kontoavtalen. Dette gjøres ved at brukeren enkelt trykker på en knapp merket "Signer kontoavtale", og deretter kan sende avtalen videre til bank.

11. Bekrefte at betaling er gjort

På lik linje med signering av kontoavtale, har systemet en løsning der brukeren kan bekrefte at betaling er gjort. Dette er enkle operasjoner som består av å klikke på en avsjekkings-boks og deretter bekrefte.

4.3 Administrative resultater

Vi har brukt Scrumban som utviklingsprosess, og ved oppstarten av bacheloroppgaven delte vi prosessen inn i sprinter og definerte hva de forskjellige sprintene skulle omhandle etterfulgt med klare mål. Videre ble delmål og oppgaver innen hver sprint overført til Trello og oppført på backloggen.

Vi kom i gang litt senere enn det vi hadde planlagt med sprint 1 som omhandlet innloggingen. Det som dro ut tiden var planleggingen - videreføring av systemutviklingsprosjektet fra høsten, presentasjoner av prosjektet, samt å finne ut hvordan vi skulle løse en sikker innlogging på applikasjonen ved å ta i bruk den eksisterende løsningen som Fixrate bruker. Når vi endelig fant en løsning som fungerte for innloggingen, ble det med en gang en fin flyt i utviklingen og framdriften av prosjektet.

Nest siste sprint i utviklingen omhandlet gjøremålslisten, der det ble brukt litt lengre tid enn det som var planlagt. Etter flere demoer med oppgavestiller og Fixrate angående denne sprinten, fikk vi kommentarer og tilbakemeldinger på endringer som de ønsket. Det resulterte i at vi brukte en uke lengre enn det vi hadde planlagt. Det ga oss mindre tid til å jobbe med Dashbordet som var tenkt som siste sprint i prosessen. For mer utfyllende informasjon om administrative resultater, se Prosjekthåndbok [32].

5 Diskusjon

Diskusjonsdelen vil ta for seg utfordringer gjennom bacheloroppgaven og reflektere over resultater gjennom systemutviklingen.

5.1 Brukertesting

Vi hadde planlagt å gjennomføre brukertester på kundene hos Fixrate etter at vi hadde fullført sprint 4 som omhandlet gjøremålslisten, men på grunn av korona-pandemien [5.4] ble ikke brukertesting aktuelt. Målet var å ha en prototype av applikasjonen som kunne brukes til testing på kundene. Det ble diskutert en eventuell løsning hvor vi gjorde testingen gjennom samhandlingsverktøyet Teams, men siden vi har lagd en standalone applikasjon, er det ikke like enkelt å få kundene til å kjøre applikasjonen på sin egen telefon lokalt. For at kunden skal kunne klare å kjøre det lokalt, krever det at de har prosjektet lokalt på sin egen datamaskin, og deretter kan starte applikasjonen med en kommando gjennom en terminal. Tilbakemeldingene som vi kunne ha fått gjennom brukertesting ville ha vært veldig nyttig for resultatet av applikasjonen med å få en oversikt over kundens behov og krav, og vi kunne ha utelukket eventuelle funksjonaliteter som manglet, var forvirrende eller som ikke var nødvendige.

5.2 Vitenskapelige resultater

5.2.1 Produkt

Innlogging med BankID og biometri

Applikasjonen tilbyr funksjoner som tar i bruk biometrisk maskinvare i telefonen og tilbyr innlogging med Bank-, Face- og TouchID. Funksjonaliteten tilbyr en bedre brukeropplevelse ved å ta i bruk kjente innloggingsmetoder som brukes i de fleste applikasjoner i dag som krever innlogging. Resultatet av innloggingen fungerer ved levering og er slik oppdragsgiver ønsket ved oppstart av bacheloroppgaven. En svakhet ved innloggingen er en noe komplisert logikk i kildekoden og den bør derfor utbedres i videre arbeid. Innloggingen fungerte i utgangspunktet ikke ved oppstart av prosjektet, og det ble det brukt veldig mye tid på å finne en løsning. Grunnen til at logikken ble så komplisert er at det var veldig mange forskjellige use-cases som måtte oppfylles, så det ble ikke prioritert å sette av ekstra tid til å utbedre den videre.

Under workshopen med utviklerteamet til Fixrate diskuterte vi eventuelle løsninger på innloggingen og det ble tatt opp mange forskjellige forslag til løsninger som vi kunne ta i bruk. Det ble stadig diskutert nye løsninger ettersom noen av de foreslåtte løsningene viste seg å ikke kunne brukes. Det skapte mye usikkerhet om hvordan innloggingen skulle være, samtidig som at vi ble oppfordret til å bestemme selv hvordan vi ønsket å implementere

løsningen. Det ble tatt opp flere teknologier og metoder for hvordan vi kunne håndtere sikker innlogging som omhandlet utveksling av nøkkelpar og kryptering. På et tidspunkt ble det foreslått og undersøkt en løsning for nøkkelpar-kryptering som de selv skulle ta i bruk i sitt system. Implementeringen av en slik løsning ble ansett som for stor og passet heller som sin egen bacheloroppgave.

Noe som kunne ha blitt gjort annerledes er at vi kunne ha forholdt oss til færre personer da vi var i planleggingsfasen, siden det var mange forskjellige løsninger på hvordan dette kunne løses fra flere av utviklerne. Noen av forslagene ble også diskutert innad i Fixrate, noe som antydet at det var usikkerhet i hvordan dette kunne løses. Det bidro til mer usikkerhet for oss og hvordan vi skulle finne ut hva som var den beste løsningen.

Den endelige løsningen som vi implementerte i applikasjonen ved levering bruker den samme innloggingssiden som brukes på nettstedet til Fixrate. Løsningen kan utbedres som nevnt tidligere, men oppfyller de kravene som ble satt av oppgavestiller.

Markedsplassen

For å vise markedsplassen i applikasjonen brukte vi, som nevnt tidligere, et WebView som viser nettstedet til Fixrate. Ved å gjøre det på denne måten kan Fixrate enkelt gjøre endringer på markedsplassen i sitt system, og disse endringene vil være med over i applikasjonen.

På markedsplassen vises det en header i toppen, der brukeren kan trykke på en meny og deretter trykke seg videre inn på de forskjellige sidene som da vises på nettstedet til Fixrate. Det er ikke tenkt at denne skal være med videre i applikasjonen. Med denne muligheten kan brukeren agere på de aller fleste funksjonene som er i systemet til Fixrate og som ikke oppleves som brukervennlige den dag i dag. Vi har gjort forsøk på å fjerne headeren, men klarte ikke å komme fram til en løsning som ga et konsistent resultat. Etter samtaler med Fixrate, ble det enighet om at det i fremtiden skulle være mulighet til å ha en parameter i URL slik at denne navigasjonsbaren ikke blir med i visningen på applikasjonen.

Nye annonser

Fra tidlig stadige hadde vi sett for oss en løsning der vi viste både gjøremålslisten og nye annonser i samme dedikerte side i applikasjonen. Både siden for nye annonser og gjøremål ble implementert i samme sprint som omhandlet gjøremålslisten. Etter demoer og visninger for oppgavestiller ble det en enighet om at vi skulle prøve å legge de to sidene i egne sider i applikasjonen. Resultatet ble mye ryddigere enn det som var planlagt fra start av. På de nye annonsene var det tenkt at de skulle fungere på lik måte som på markedsplassen, der de ble vist fram som kort og at brukeren kunne trykke på hele kortet for å bli dratt videre inn til markedsplassen. Løsningen vi endte opp med ved levering var at det ble lagt inn en knapp som brukeren kan trykke på istedenfor at hele kortet er trykkbart. Dette ble gjort fordi etter tilbakemeldinger fra både oppgavestiller og andre i Fixrate, så var det vanskelig å se eller forstå, hvilke aspekter av de nye annonsene som var trykkbare.

Gjøremålsliste
Gjøremålslisten var etter innloggingen den sprinten som vi brukte lengst tid på å fullføre. Under denne sprinten gikk vi gjennom flere iterasjoner med endringer og tilbakemeldinger fra oppgavestiller.

Vi fikk implementert det som var ønskelig fra oppgavestiller av funksjonalitet. Når det kommer til UX og design er det mer å gå på når det kommer til denne implementeringen i applikasjonen. Det var utfordringer med å få til et godt design på gjøremålslisten, og samtidig få det til å bli sømløst med det eksisterende systemet. Siden denne delen er et nytt element som blir presentert for brukeren, hadde Fixrate og oppgavestiller flere innspill å komme med når det gjaldt design og funksjonalitet. Først og fremst ble funksjonalitet prioritert over design. På grunn av måten vi arbeidet på, var det enklere å få hyppigere tilbakemeldinger på det resultatet vi hadde produsert, og enklere å rette opp i eventuelle feil og mangler.

Dashboard
Løsningen som vi endte opp med ved leveringstid var ikke planlagt fra begynnelsen av. Vi hadde planlagt å utforme dashbordet som en widget som kunne vises på en låst skjerm på telefonen. Vi begynte med å utforme dashbordet på den siste sprinten i utviklingsfasen. Etter diskusjoner med oppdragsgiver og Simen Bjørkhaug på utviklerteamet, ble det besluttet at vi ikke skulle bruke mye tid på dette. Som nevnt i kapittel 4.1.1 ble det nevnt fremtidige oppdateringer i iOS med større fokus på widgets. Derfor er det stor grunn til å tro at man må gjøre store endringer på utviklingen rundt en eventuell widget når denne oppdateringen kommer og det ble derfor satt på vent inntil videre. Ingenting er bekreftet av Apple enda, men det ventes nærmere informasjon rundt dette i juni 2020.

Pushvarsler
Pushvarsler var en av hovedfunksjonene Fixrate ønsket at vi skulle implementere i applikasjonen. Formålet var å prøve å effektivisere hele prosessen mellom bank og innskyter når en innskyter starter en bestilling.

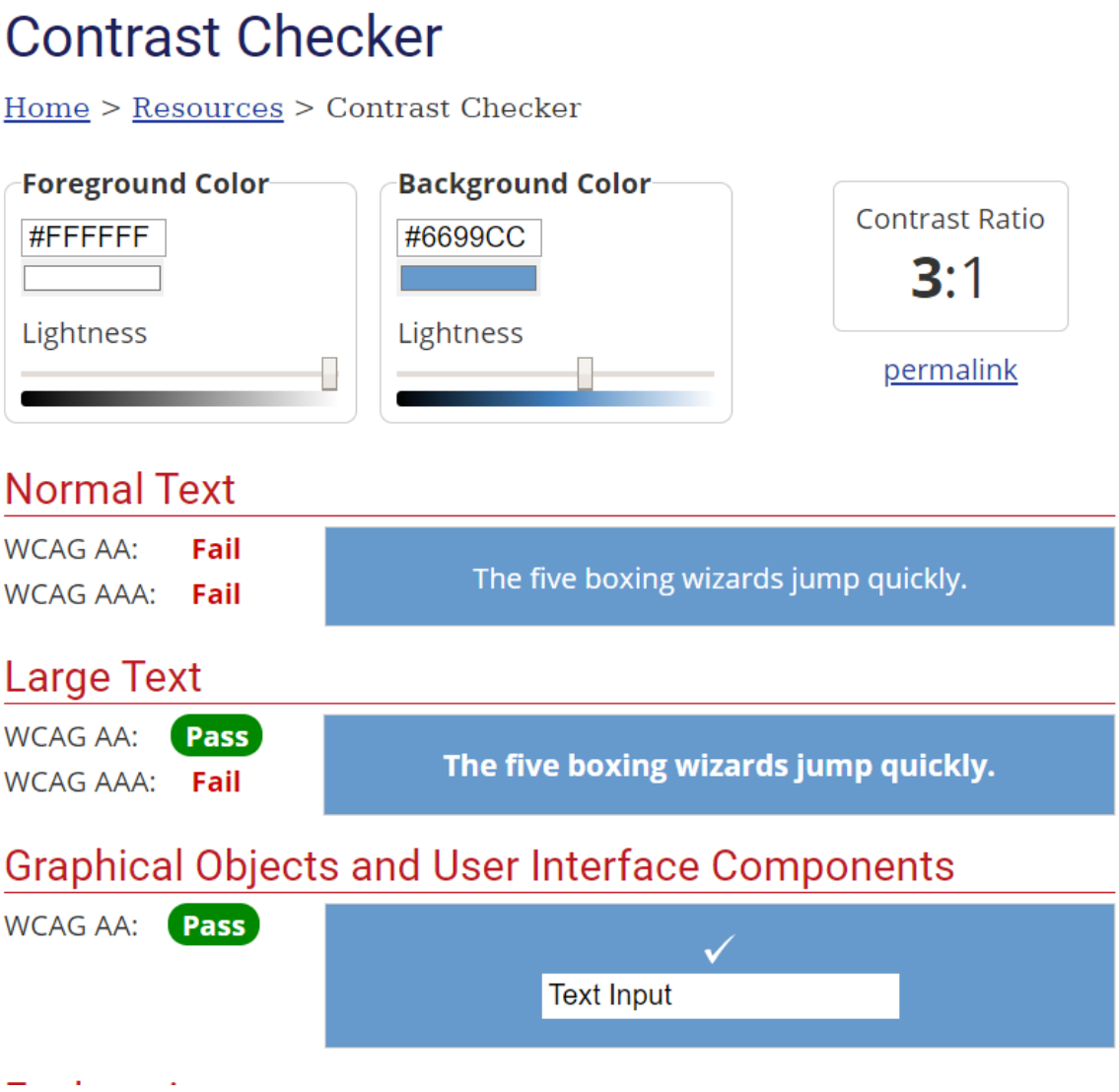
Under denne sprinten ble det arbeidet mye med å finne de rette eventene som skulle svare til et pushvarsel hos bruker, ikke minst innholdet for hvert pushvarsel lå det mye arbeid bak. Vi hadde fått tilgang til en testmail-server som Fixrate bruker i forbindelse med å sende ut mailvarsling for nye hendelser på nettstedet. Denne brukte vi for å sammenligne det tekstlige innholdet vi skulle ha i pushvarslene med det som blir sendt ut som e-post til brukerne. Når innholdet var definert brukte vi databasen til Fixrate for å hente ut de verdiene som vi skulle bruke til systemet. I videre arbeid kan man argumentere for at det er hensiktsmessig å sette opp sin egen tjeneste for utsending av pushvarsler. Dette kan være en god ting siden tjenesten som er i bruk nå kan gå over til en betalingsplan i fremtiden.

Det ble implementert mulighet for at brukeren kan agere på pushvarslene som dukker opp på telefonen. Hvis brukerne trykker på varselet blir de tatt videre inn i applikasjonen til den dedikerte siden som pushvarselet omhandler.

På android har vi oppdaget at mottakelse av pushvarsler ikke fungerer helt som det skal. Vi vet muligens hvor feilen ligger, men i mangel på tid til å utbedre problemet, må dette heller utbedres videre på et senere tidspunkt. Problemet ble oppdaget litt for sent grunnet manglet tilgang til telefoner som kjører på Android og korona-situasjonen som gjorde det vanskeligere å låne telefoner av venner eller bekjente [kap 5.4].

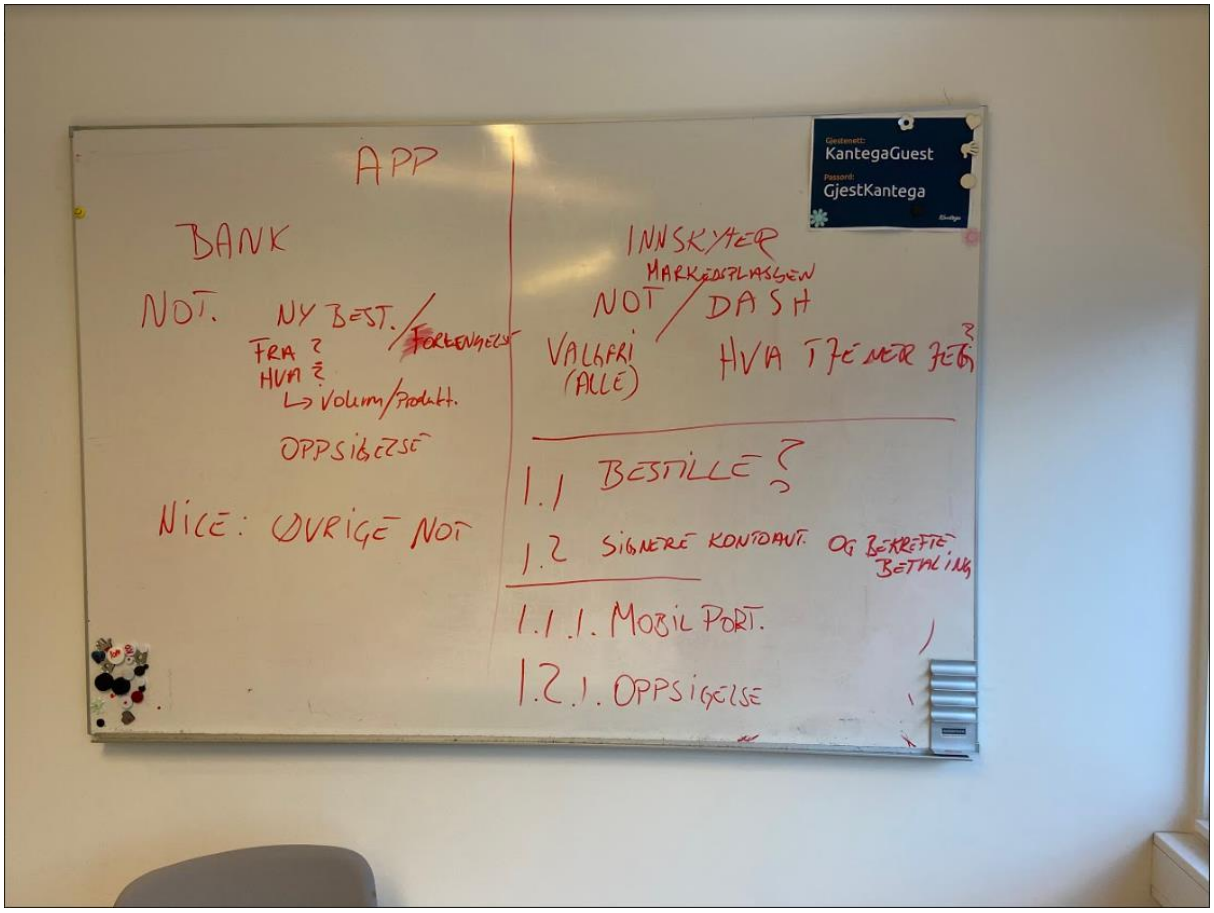
5.2.2 Design

Utfordringer ved å lage en applikasjon til et eksisterende system er at man i veldig mange tilfeller er låst til eksisterende løsninger og design. Etter undersøkelser har vi funnet ut at deler av designet som har blitt overført fra det eksisterende nettstedet til applikasjonen ikke oppfyller de universelle utformingskravene som er anbefalt når det kommer til kontraster mellom bakgrunn og tekst. Vi testet blant annet kontrastene på tekst og bakgrunn som vist i figur 19. Resultatene under viser at teksten som viser organisasjonsnavn og bakgrunnsfargen scorer dårlig ifht. hva som er anbefalte retningslinjer. Kravet for den visuelle presentasjonen av tekst fra WCAG skal ha en kontrast rate på 4.5:1 [41]. Vi valgte likevel å forholde oss til fargevalget som var gitt på Fixrate sitt nettsted fordi vi ville holde oss til designprinsippet som omhandler Consistency [2.4] - i dette tilfellet vektlagt å ikke introdusere nye designelementer. Vi valgte også å gjøre dette for å skape et mest mulig sømløst design mellom nettsted og applikasjon.



Figur 26: Skjermbilde av resultater fra kontrastsjekk [42]

5.3 Ingeniørfaglige resultater



Figur 27: Oppstartsmøte med oppgavestiller

Under oppstartsmøtet med oppgavestiller ble det definert hva applikasjonen skulle innholde ved leveringstid. Versjon 1.0 var det som var forventet at vi skulle klare å levere og øvrige versjoner av applikasjonen ble sett på som "nice to have". Målene som er satt opp i visjonsdokumentet, og beskrevet under kap 4.2.1 er definert ut fra bildet vist over. Flere av de funksjonalitetene og målene som er satt er "nice to have" funksjoner som vi har implementert i applikasjonen. Ut fra bildet ser vi at det er definert en versjon 1.1.1 Mobil portefølje. Dette er en av de få tingene som vi ikke har fått implementert. Under siste møte med oppgavestiller ble det tatt opp ønsker og muligheter for å få til en løsning for mobil portefølje, og vi kom til enighet om at hvis det var tid til overs, så kunne vi se om vi fikk implementert dette. Under oppstartsmøtet ble det klart at applikasjonen ikke skulle være en erstatning for det eksisterende systemet, men heller et tilskudd. Ved å implementere en mobil portefølje i applikasjonen, nærmer vi oss heller en applikasjon som kan erstatte nettstedet for en bruker som er innskyter. Tatt dette i betraktning har vi besluttet at vi ikke skulle gjennomføre dette målet for å begrense mulighetene en kan gjøre gjennom applikasjonen.

5.4 Utviklingsmetodikk og arbeidsprosess

Vi valgte å benytte oss av utviklingsmetodikken ScrumBan og den digitale løsningen Trello for å holde oversikt over sprinter og arbeidsoppgaver. Samtidig oppsto det en helt unik og spesiell situasjon for alle i Norge. Tiltakene som ble innført for å begrense smittespredningen av koronaviruset [43] gjorde at det ble innført hjemmekontor for alle som hadde mulighet til å jobbe hjemmefra. NTNU stengte dørene for studenter og regler for hvordan en skal oppholde seg blant andre ble innført. Dette medførte at vi ble nødt til å jobbe hjemmefra resten av bacheloroppgaven.

Ettersom vi hadde gode arbeidsrutiner og var vant til å jobbe sammen enten på skolen eller på kontor med tilgang til fysiske tavler for problemløsning ble det utfordringer med å opprettholde den samme gode arbeidsflyten da vi ble nødt til å sitte hjemme hver for oss.

For å få et tilnærmet likt godt samarbeid tok vi i bruk flere digitale samhandlingsverktøy, slik at det skulle bli enklere for hver enkelt å henge med og forstå hva som ble gjort i prosjektet. Denne situasjonen oppstod halvveis ut i bacheloroppgaven, så måten vi arbeidet på og hvordan vi kommuniserte med hverandre, veileder og oppgavestiller ble gjort om fra personlige møter til videomøter gjennom digitale samhandlingsverktøy.

Fordelen ved bruk av ScrumBan var at vi hele tiden hadde mulighet til å legge til oppgaver underveis i sprintene våre. Dette var veldig nyttig i de tilfellene hvor vi kom på oppgaver som vi trengte å løse underveis i sprinten og ble oftere tilfelle når vi ble sittende og jobbe hjemmefra.

Ulempen ved ScrumBan var at vi i starten kanskje ikke var like flinke å markere hvilke oppgaver vi jobbet med. Dette var mest et resultat av mangel på erfaring og bruk av Trello sin tavleløsning. Dette medførte litt ineffektivt arbeid i noen tilfeller der en av oss hadde utført en oppgave, mens den andre jobbet videre på samme oppgave. Problemet rundt tildelingen av oppgaver ble løst etterhvert som vi oppdaget det og vi ble veldig flinke på å tildele oppgaver til oss selv slik at den andre kunne se hva som ble arbeidet med.

5.5 Helhetlig systemperspektiv

Applikasjonen som har blitt utviklet under bacheloroppgaven er ment som et enkelt og intuitivt tilskudd til Fixrates eksisterende nettsted og kunder. For bruk av biometri i applikasjonen, kreves det at mobilen har maskinvare som støtter dette, men alle øvrige funksjonaliteter i applikasjonen er likevel tilgjengelig. Applikasjonen har gjennom flere møter og tilbakemeldinger fått samme design og elementer som kundene enkelt kjenner igjen og har et forhold til gjennom det eksisterende nettstedet. Dette gjør derfor at brukerne kan forstå å ta i bruk applikasjonen veldig enkelt.

Som nevnt tidligere viser vi innhold fra nettstedet til Fixrate direkte i applikasjonen. Dette betyr at når innholdet på nettstedet endrer seg, vil det endres i applikasjonen samtidig.

Økonomisk sett er dette en fordel siden man da slipper å bruke tid på å endre kildekoden i både applikasjon og nettsted for å opprettholde samme design. Resultat av en slik løsning er at brukere som allerede er vant til å bruke nettstedet og tar applikasjonen i bruk, ikke trenger noen form for opplæring. Dette betyr mindre vedlikehold og kundesupport, noe som er en klar fordel når man er et mindre utviklingsteam med begrensede ressurser.

5.6 Gruppearbeidet

Ettersom at vi begge har jobbet sammen siden høstprosjektet med samme utgangspunkt, ferdigheter og et ønske om å utvikle seg selv i faget, har det derfor fungert utmerket å fortsette samarbeidet. Ved oppstarten av bacheloroppgaven hadde vi allerede arbeidet i sammen og var derfor godt kjent med både hverandres ferdigheter og arbeidsmoral. Siden ferdighetsnivået har vært likt gjennom hele utviklingen har vi begge derfor stått på og gjort nødvendig arbeid for å få gjennomført oppgaven. Vi har ikke støtt på noen uenigheter under utviklingen og alltid hatt en åpen dialog om hvordan ting skal gjennomføres.

Vi startet denne oppgaven i høst for å utfordre oss selv faglig, bygge selvtillit og få kunnskap rundt applikasjon- og webutvikling. Hele perioden har derfor vært veldig lærerik og gitt oss veldig mye utbytte i form av ny kunnskap og erfaringer fra arbeidslivet som vi kan ta med oss videre.

6 Konklusjon

Problemstillingen presentert i oppgaven er som følger:

Hvordan lage et nytt system for mobile plattformer som både baserer seg på og sameksisterer med et allerede eksisterende system?

- *Hvordan sikre brukerne en sikker innlogging med gjenbruk av løsning fra eksisterende system til nytt system?*
- *Hvilke designprinsipper må praktiseres for å ikke introdusere nye designelementer i systemet?*

Den største utfordringen med problemstillingen har vært å holde kontroll på all data som allerede eksisterer og hvordan dette brukes i dag. Store deler av systemutviklingen har derfor blitt brukt på å kartlegge og forstå hvordan ting henger sammen i det eksisterende nettstedet, hvordan data endrer seg når brukere gjør handlinger i systemet og ikke minst forstå konseptene rundt systemet og hvordan det brukes av kundene til Fixrate. Det har derfor vært veldig viktig å hele tiden ha god kontroll på og forståelse for, siden det eksisterende systemet skal påvirke applikasjonen vi har utviklet. Vi har under hele utviklingen hatt tilgang til nettstedet som er laget for testing slik at vi kan klikke rundt og bli godt kjent med alle funksjonene i systemet og registrere når data endrer seg. Siden vi har arbeidet til dels på kontoret til Fixrate har de vært behjelpelige og svart på alle spørsmål vedrørende systemet deres.

Utfordringer ved å lage en applikasjon til et eksisterende system er at man i veldig mange tilfeller er låst til eksisterende løsninger og design. Som nevnt i rapporten har vi tidligere hatt problemer med innloggingen til applikasjonen. Hadde dette vært en frittstående applikasjon, som ikke har noen sammenheng med et annet system, kunne vi fritt valgt hvordan innloggingen skulle ha vært. Dette hadde nok spart oss veldig mye tid både i denne oppgaven og i prosjektet vi hadde i forkant av bacheloroppgaven, siden innloggingen har vært en stor utfordring. Det samme gjelder designet til applikasjonen. Her har vi vært låst til design siden det er vanlig og regnes som “best practice” å overføre designet fra eksisterende nettsted til en tilhørende applikasjon slik at det oppfattes som samme produkt for brukeren.

Applikasjonen som har blitt utviklet er fungerende, mottar pushvarsler og er av samme design og elementer som brukerne allerede er vant til på nettstedet. Vi kan derfor si at vi har lyktes med å utvikle et nytt system for mobile plattformer som baserer seg på og sameksisterer med Fixrates eksisterende system.

6.1 Videre arbeid

Applikasjonen består av alle funksjonelle krav gitt av oppgavestiller fra start og har blitt formet underveis etter tilbakemeldinger. Videre arbeid på applikasjonen krever at Android-brukere blir satt i fokus. Siden applikasjonen hovedsaklig har blitt kontinuerlig testet på iPhone og det har vært mangel på tilgang til en mobiltelefon som kjører Android har det vært vanskelig å teste applikasjonen på dette operativsystemet. Funksjoner som pushvarsler og biometrisk innlogging har vært særlig vanskelig å teste på Android. Vi har virtuelle telefoner som kan kjøres på datamaskin, men for å teste pushvarsler og biometrisk innlogging kreves det en mobiltelefon. Dette er fordi den biometriske innloggingen krever ekte maskinvare, og pushvarsler krever en unik identifikasjon gitt av telefonen. Videre arbeid på applikasjonen krever derfor et større fokus og testing på en telefon som kjører Android.

Av fungerende funksjonalitet i applikasjonen skulle vi gjerne jobbet mer med logikken rundt innloggingen og måten biometrisk innlogging blir tatt i bruk. Som nevnt tidligere kan logikken her være vanskelig å forstå å sette seg inn uten å ha jobbet med den selv. Derfor skulle vi gjerne hatt mulighet til å ta oss mer tid til å se på det for å komme frem til en bedre løsning.

Sidene i applikasjonen som viser innhold direkte fra Fixrates nettsted krever videre arbeid. Som nevnt tidligere i rapporten viser eksempelvis markedsplassen en header som muliggjør navigasjon på nettsiden som vises. Dette er ikke ønsket i applikasjonen og bør derfor gjøres noe med. Videre arbeid kan være at URLen til nettstedet som vises får sitt eget parameter slik at Fixrate kan velge å vise en egen nettside uten header i applikasjonen. Dette gjelder samtlige sider hvor det vises innhold fra nettstedet i applikasjonen.

Videre skulle vi gjerne videreutviklet dashboardet til brukerne da oppgavestiller kom med flere ønsker mot slutten bacheloroppgaven. På grunn av lite tid og prioritering av dokumentasjon så vi oss nødt til å nedprioritere det. Her skulle vi lagt inn en liste over all ordre brukeren har, noe som kan forstås som en liten portefølje i tillegg til ordre med forfallsdatoer.

All utsending av pushvarsler foregår i RabbitMQ serveren som er satt opp. Hver sending bruker en unik identifikasjon som mottaker. Når pushvarslene har blitt sendt ut, mottar RabbitMQen kvittering fra APIet disse sendes til. Alle feil som forekommer under utsendingen av pushvarsler står i kvitteringen. Videre arbeid bør ta for seg nærmere håndtering av kvitteringen slik at de rette prosedyrene vil settes i gang ved eventuelle feil. I dag fjerner systemet ugyldige ider fra databasen hvis det forekommer. Eksempler på håndtering som bør implemeneteres er håndtering av meldinger som er for store eller om man prøver å sende for mange pushvarsler til samme bruker.

En stor del av utviklingen og resultatet av et produkt avhenger av tilbakemeldinger fra brukertester. Disse fikk vi ikke gjennomført som planlagt på grunn av korona-pandemien som satte restriksjoner ved at vi var nødt til å holde oss hjemme. Videre arbeid bør derfor inneholde brukertester hvor applikasjonen testes grundig på kunder for å få tilbakemeldinger på det som har blitt utviklet. Videre kan man definere nye arbeidsoppgaver etter at brukertester har blitt gjennomført, og man har fått konkrete tilbakemeldinger på hva som oppleves som bra eller dårlig.

Referanser

[1] Visjonsdokument. Se vedlagt zip-fil

[2] Høstprosjekt: Hovedrapport. Se vedlagt zip-fil

[3] What is continous integration https://aws.amazon.com/devops/continuous-integration/ (Besøkt Mai. 2020)

[4] Getting started - About Version Control https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control (Besøkt Mai. 2020)

[5] Designprinsipper https://www.uio.no/studier/emner/matnat/ifi/INF1500/h15/plenumstimer/inf1500-h15-uke-8---designprinsipper-del-1.pdf (Besøkt Mai. 2020)

[6] Signicat Support and Security https://www.signicat.com/about/support-and-security/ (Besøkt Mai. 2020)

[7] How SAML authetication works https://auth0.com/blog/how-saml-authentication-works/ (Besøkt Mai. 2020)

[8] Om avansert Face ID-teknologi https://support.apple.com/no-no/HT208108

(Besøkt Mai. 2020)

[9] Brukertesting https://www.northernbeat.no/tjenester/brukertesting/ (Besøkt Mai. 2020)

[10] Fem råd for gjennomføring av brukertest https://blogg.kantega.no/content/1351/Fem-rad-for-gjennomforing-av-brukertest

(Besøkt Mai. 2020)

[11] React-Native https://facebook.github.io/react-native/ (Besøkt Mai. 2020)

[12] React-Native (Språk og rammeverk) https://radar.bekk.no/tech2018/sprak-og-rammeverk/react-native (Besøkt Mai. 2020)

[13] Balsamiq Wireframes https://balsamiq.com/wireframes/ (Besøkt Mai. 2020)

[14] Expo Documentation: Introduction to Expo https://docs.expo.io/ (Besøkt Mai. 2020)

[15] Expo Documentation: Features https://expo.io/features (Besøkt Mai. 2020)

[16] Messaging that just works - RabbitMQ https://www.rabbitmq.com/ (Besøkt Mai. 2020)

[17] Redux - A predictable state container for JavaScript apps. https://redux.js.org/ (Besøkt Mai. 2020)

[18] GitLab Docs https://docs.gitlab.com/ee/topics/git/index.html (Besøkt Mai. 2020)

[19] Trello https://www.kompetansebroen.no/tools/trello/ (Besøkt Mai. 2020)

[20] Jest - Delightful JavaScript Testing https://jestjs.io/ (Besøkt Mai. 2020)

[21] Introduction - Enzyme https://enzymejs.github.io/enzyme/ (Besøkt Mai. 2020)

[22] Unit testing with JUnit https://www.vogella.com/tutorials/JUnit/article.html (Besøkt Mai. 2020)

[23] What is Babel? - Babel https://babeljs.io/docs/en/ (Besøkt Mai. 2020)

[24] About Node.js https://nodejs.org/en/about/ (Besøkt Mai. 2020)

[25] Express - Node.js web application framework https://expressjs.com/ (Besøkt Mai. 2020)

[26] MySQL 8.0 Reference Manual https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html (Besøkt Mai. 2020)

[27] What is a Container? - Docker https://www.docker.com/resources/what-container (Besøkt Mai. 2020)

[28] What is Kubernetes? https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/#why-you-need-kubernetes-and-what-can-it-do (Besøkt Mai. 2020)

[29] Droplets on DigitalOcean https://www.digitalocean.com/products/droplets/

(Besøkt Mai. 2020)

[30] Systemdokumentasjon. Se vedlagt zip-fil

[31] iOS 14: All the Rumored New Features https://www.macrumors.com/roundup/ios-14/ (Besøkt Mai. 2020)

[32] Prosjekthåndbok. Se vedlagt zip-fil

[33] Sprint_1_Innlogging. Se vedlagt zip-fil

[34] Signicat https://www.signicat.com/no/ (Besøkt Mai. 2020)

[35] Instagram https://www.instagram.com/?hl=nb (Besøkt Mai. 2020)

[36] Discord https://discord.com/ (Besøkt Mai. 2020)

[37] Facebook https://www.facebook.com/ (Besøkt Mai. 2020)

[38] Ubuntu https://ubuntu.com/ (Besøkt Mai. 2020)

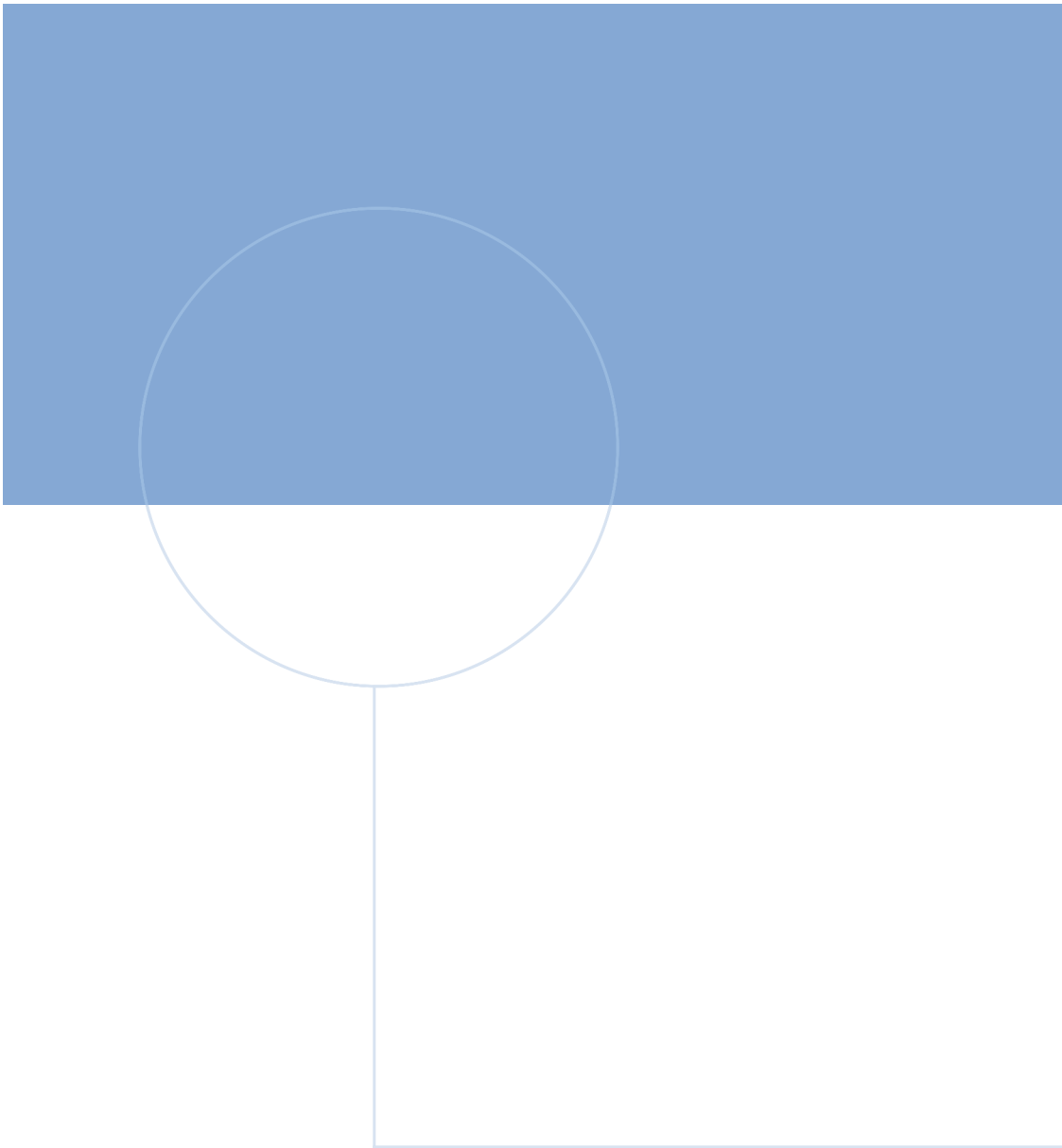
[39] Wireframes. Se vedlagt zip-mappe

[40] Apple https://www.apple.com/no/ (Besøkt Mai. 2020)

[41] WCAG Contrast (Minumum) -Level AA
https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=143#contrast-minimum
(Besøkt Mai. 2020)

[42] Contrast Checker https://webaim.org/resources/contrastchecker/ (Besøkt Mai. 2020)

[43] Koronavirus-pandemien i 2020 https://sml.snl.no/koronavirus-pandemien_i_2020
(Besøkt Mai. 2020)



NTNU

Kunnskap for en bedre verden