Kjetil Wilhelmsen Helgås
Yngve Jarand Kjelgum Hestem
Jørgen Jærnes

FRIDA App

Bacheloroppgave i Programmering
Veileder: Tom Røise
Mai 2019

Bacheloroppgave

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

NTNU
Kunnskap for en bedre verden

Kjetil Wilhelmsen Helgås
Yngve Jarand Kjelgum Hestem
Jørgen Jærnes

FRIDA App

Bacheloroppgave i Programmering
Veileder: Tom Røise
Mai 2019

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

NTNU
Kunnskap for en bedre verden

Norwegian University of
Science and Technology

FRIDA App

Forfattere
Kjetil Wilhelmsen Helgås
Yngve Jarand Kjelgum Hestem
Jørgen Jærnes

Bachelor i programmering[spill|apps]
20 ECTS
Institute for Datateknikk og Informatikk
Norges teknisk-naturvitenskapelige universitet,

20.05.2019

Veileder Tom Røise

Sammendrag av Bacheloroppgaven

Tittel:	**FRIDA App**
Dato:	20.05.2019
Deltakere:	Kjetil Wilhelmsen Helgås Yngve Jarand Kjelgum Hestem Jørgen Jærnes
Veiledere:	Tom Røise
Oppdragsgiver:	EVRY Norge AS
Kontaktperson:	Johnny Bjørnstad, johnny.bjornstad@evry.com, +47 913 08 316
Nøkkelord:	Frivillighet, Android, iOS, Oracle JET
Antall sider:	65
Antall vedlegg:	3
Tilgjengelighet:	Åpen

Sammendrag: FRIDA appen skal erstatte og forbedre en allerede eksisterende app og gjøre det lettere for frivillige og ansatte å bruke FRIDA i hverdagen. Systemet FRIDA består i forkant av oppgaven av flere moduler, blant annet en database, et SMS-varslingssystem og en nettside. Oppgaven må tilpasses disse. Appen er bygget med Oracle JET, som tar i mot HTML og Javascript og viser dette gjennom viser dette gjennom et WebView til både Android og iOS enheter. Det er viktig at appen er lett tilgjengelig for en bred målgruppe og dermed tar hensyn til universell utforming. EVRY vurderer å ta i bruk appen og klargjøre den for publisering.

Summary of Graduate Project

Title:	**FRIDA App**
Date:	20.05.2019
Authors:	Kjetil Wilhelmsen Helgås Yngve Jarand Kjelgum Hestem Jørgen Jærnes
Supervisor:	Tom Røise
Employer:	EVRY Norge AS
Contact Person:	Johnny Bjørnstad, johnny.bjornstad@evry.com, +47 913 08 316
Keywords:	Volunteer, Android, iOS, Oracle JET
Pages:	65
Attachments:	3
Availability:	Open

Abstract: The FRIDA app is made to replace and improve an already existing app, which allows volunteers and employees to use the FRIDA system everywhere. Prior to the project, FRIDA consists of several modules, including a database, an SMS-notification service and a website. The project will have to adapt to these modules. The app is built with Oracle JET, which uses HTML and Javascript to display a view through WebView, and works on both Android and iOS. It is important that the app is accessible to its wide userbase, and takes into account the rules of Universal Design. EVRY is considering using the app and readying it for publication.

Forord

Oppgaven ble presentert som en åpen oppgave, noe som gav gruppen mulighet til å være med å definere innholdet. Den minnet om noe vi hadde gjort, og trivdes med å gjøre tidligere. Ettersom alle gruppens medlemmer har valgt retningen applikasjonsutvikling, passet den godt innenfor hva vi hadde lært, i tillegg til å by på nye teknologier og utfordringer.

Vi vil gjerne takke alle som var med å hjelpe i EVRY Innlandet, hvorav vi gir en ekstra takk til Johnny Bjørnstad, Merete Myren og Guro Storlien, som alle fulgte oss gjennom hele prosjektet. Vi vil også takke Frivillighetssentralen i Hamar, som var med på å teste og gi tilbakemeldinger på applikasjonen. Til slutt vil vi takke vår veileder, Tom Røise, for god veiledning og konstruktive tilbakemeldinger gjennom hele prosjektet.

Gjøvik, mai 2019

Kjetil Wilhelmsen Helgås Yngve Jarand Kjelgum Hestem Jørgen Jærnes

Innhold

Figurer

Tabeller

Listings

1 Introduksjon

1.1 Problemområde og oppgavebeskrivelse

FRIDA, kort for Frivillighetssentralenes Data Administrasjonsverktøy, er et system utviklet av EVRY AS som skal effektivisere og forenkle frivillighetsarbeid. Systemet benyttes primært gjennom en nettside med en nettleser. I forkant av dette prosjektet fantes det også en mobilapplikasjon for FRIDA, men denne hadde en begrenset brukergruppe og funksjonalitet. Oppgaven består dermed av å lage en ny applikasjon for FRIDA. Første utkast av oppgaven besto av å lage en ny app for frivillige med noe utvidet funksjonalitet, men dette kunne utvides dersom gruppen skulle bli ferdige med dette tidlig i prosjektet. Oppgaven ble etterhvert utvidet til å inkludere en egen side for ansatte og administratorer.

Appen skal gjøre det lett for frivillige å administrere brukeren sin raskt og hvor som helst, da den nåværende nettsiden er bygget for PC og er dermed vanskelig å navigere på telefon. En kan trekke ut fire funksjonaliteter som de mest sentrale i appen, da disse er noe en bruker ofte vil trenge å gjøre uten tilgang til PC. Disse er å akseptere og avslå forespørsler, samt melde interesse til andre aktiviteter, redigere timer på tidligere tjenester dersom det skulle vare kortere eller lenger enn oppført, melde utilgjengelig både ukentlig og over en periode og en kalender med oversikt over alt brukeren kan eller skal på. Utover dette skal brukeren også kunne endre innstillinger og opplysninger, samt vise et FRIDA ID-kort til tjenestemottaker når han/hun er på et oppdrag. Det finnes mer funksjonalitet på nettside versjonen av FRIDA, men da appen er laget primært for at brukere kan gjøre hyppige oppgaver kjapt på farta er det viktigere at appen holdes enkel og oversiktlig.

1.2 Målgruppe

Denne rapportens målgruppe er primært sensor, veileder og arbeidsgiver. Den er skrevet med forutsetningen om at leseren har samme tekniske bakgrunn som medlemmene av gruppen eller mer.

FRIDA har brukere i alle aldre fra ung voksen til pensjonist. Appen vil dermed tas i bruk av både personer som aktivt bruker smarttelefon i hverdagen og personer som bare bruker den for å ringe og å sende meldinger. I tillegg antar vi at mange vil bruke appen i situasjoner hvor det er hensiktsmessig å kjapt kunne aksessere og utføre de fire hovedfunksjonene. Administrator-delen av applikasjonen er også laget for å kunne brukes i farta, men her antas det at brukerne enten får opplæring eller bruker tid på å sette seg inn i applikasjonen, da de skal bruke den i jobbsammenheng.

Selv om FRIDA primært er i bruk hos frivillighetssentraler, kan det også brukes av andre organisasjoner som arbeider med frivillighet. Røde Kors er et eksempel på en ekstern organisasjon som også tar i bruk systemet. Denne oppgaven er primært fokusert på frivillighetssentraler som målgruppe.

1.3 Bakgrunn og kompetanse

Alle gruppens medlemmer studerer Programmering og har ved valg av studieretning valgt Applikasjon. Vår faglige bakgrunn er dermed basert på 5 semestre på NTNU Gjøvik, hvor vi primært har brukt språkene C++, Java, HTML, CSS, Javascript og PHP. Gjennom studiene har vi utviklet nettsider med både PHP og Javascript, samt Android-mobilapplikasjoner med Java. Ingen av gruppens medlemmer har tidligere erfaring med Oracle JET eller andre liknende rammeverk. Vi har heller ikke brukt REST APIer til å kommunisere med databasen før, slik at dette var også noe alle måtte lære seg før vi kunne begynne.

1.4 Arbeidsmetodikk og gjennomføring

Hver tirsdag reiste gruppen til EVRYs kontor. Der ble det ofte holdt et kort møte med oppdragsgiver, enten relatert til Scrum sprinten eller andre temaer gruppen eller oppdragsgiver ville ta opp. Onsdager hadde gruppen møte med veileder kl 10:30, og møtte en halv time før for å lage en dagsorden for møtet. Resten av dagene møtte gruppen i et reservert grupperom og jobbet fra 09:00 til 16:00, med unntak av mandager som ble holdt av til andre fag om det trengtes. Trello ble benyttet som scrum board, da dette kunne integreres med repositoriet i Bitbucket.

1.4.1 Utviklingsmodell

I starten av planleggingsfasen skulle gruppen velge hvilken utviklingsmodell som skulle brukes gjennom prosjektet. De første alternativene som ble trukket frem var Scrum og Waterfall modellen. Gruppen konkluderte med at selv om vi viste hvordan sluttproduktet skulle se ut, var det stor sjanse for at noe kunne komme til å endre seg eller bli forsinket i løpet av prosjektet. Dette var spesielt aktuelt da vi skulle bruke et nytt rammeverk og jobbe med en ny type database. Det kunne derfor bli vanskelig å håndtere endringer om vi brukte Waterfall modellen.

Da vi hadde valgt å bruke en mer smidig utviklingsmodell, ble eXtreme Programming trukket frem som et mulig alternativ. eXtreme Programming bruker i likhet med Scrum flere korte utviklingsperioder, som gjør det enklere å håndtere endringer i prosjektet underveis. Parprogrammering står også sentralt i eXtreme Programming, og da gruppen kun består av tre personer vil dette være vanskelig og ineffektivt. Gruppen endte dermed opp med å velge Scrum, noe alle gruppens medlemmer var kjent med fra tidligere prosjekter. Sammen med oppdragsgiver ble det laget en Product Backlog som inneholdt alt som skulle være med i frivillig-delen av applikasjonen. Gruppen inndelte deretter Product Backlogen inn i grove Sprint Backloger, hvor hver sprint var 14 dager lang. Det ble holdt et Sprint Planning møte i starten av hver sprint og et Sprint Review møte i slutten. Noen av møtene ble kun holdt internt i gruppen, da det ikke var alltid det passet å ha dem med oppdragsgiver. Hver arbeidsdag startet også med et Daily Scrum møte, hvor alle gruppens medlemmer fortalte hva de jobbet med og hvordan de lå an. Under hver sprint var det bare Scrum Master som hadde kontakt med oppdragsgiver.

1.4.2 Prosjektorganisering og roller

Ingen av gruppens medlemmer hadde spesielle utviklingsroller, da alle har samme kompetanse. Rollene i prosjektet var som følger:

- Oppdragsgiver: EVRY, Johnny Bjørnstad

- Veileder: Tom Røise
- Gruppeleder: Kjetil Wilhelmsen Helgås
- Kontaktansvarlig og Scrum Master: Yngve Jarand Kjelgum Hestem
- Sekretær: Jørgen Jærnes

1.5 Organisering av rapporten

1.5.1 Kapitteloppsummering

Introduksjon

Dette er en kort introduksjon om oppgaven. Den inneholder også informasjon om denne rapporten.

Kravspesifikasjon

Her kommer en beskrivelse av krav og ønsker som har kommet opp i løpet av Scrum-prosessen som ønskes å bli implementert. Det kommer også en beskrivelse av nåværende app og databasen slik at man kan forstå litt av hva vi skal forbedre og bruke i prosessen.

Programvarearkitektur og design

Her forklares hvilke utviklingsverktøy som ble brukt og prosessen ved valg av programmeringsspråk. Vi beskriver også valg ang. brukergrensesnittet og om universell utforming.

Utviklingsprosess

I dette kapittelet forklares hvordan man har arbeidet i dette semesteret. I tillegg forklares hvordan gruppen har tenkt ang. sikkerhet og andre ting som enten var problematiske eller måtte tenkes på i løpet av utviklingsprosessen.

Implementasjon

Her forklares funksjonalitet for de forskjellige sidene i appen. Med dette menes hovedsaklig tanker rundt teknologien bak, men noe design kan bli nevnt om det underbygger de tekniske valgene. Vi nevner i tillegg generelle valg under utviklingsprosessen slik som hvorfor vi valgte en funksjon fremfor en annen. Gruppen snakker også om forskjeller og problemer som kan oppstå mellom ulike platformer som Android og iOS.

Distrubusjon

Her forklares testdatabasen og produksjonsmiljø nærmere. I tillegg nevnes ting som må gjøres for å sette appen ut i produksjon.

Testing og tilbbakemeldinger fra brukere

Her skrives det om testen vi fikk gjort i løpet av utviklingsprosessen hos Hamar frivillighetssentral. Der skrives det om hele prosessen, hva vi tenkte og resultatene som kom ut av testene. I tillegg til å nevne om evt. problemer gruppen hadde.

Diskusjon

Her diskuterer vi tanker vi hadde rundt oppgaven og prosessen. Dette innebærer nærmere tanker rundt valg av språk, som vi beskrev i kapittel 3. Vi nevner også hva gruppen har lært og snakker mer om hva vi tenkte om prosessen og videre atbeid.

Konklusjon

Her konkluderer vi hele oppgaven med noen siste tanker rundt prosjektet og hva vi har lært.

1.5.2 Terminologi

I rapporten brukes det spesiell terminologi på aktører og komponenter i systemer. Listen under beskriver disse:

- Tjeneste: felles betegnelse på alle tjenestene en frivillig kan delta på
- Oppdrag: forespørsel sendt ut til frivillige basert på interesser som kan aksepteres eller avslås
- Aktivitet: frivillige kan melde interesse om å delta, arrangøren vil akseptere så mange som trengs
- Bruker: generell bruker av systemet, brukes om både frivillig og administrator
- Administrator: ansatt i en organisasjon som bruker FRIDA

1.5.3 Layout

Gruppen har valgt å bruke rapportmalen som ble gitt av NTNU[4] som utgangspunkt for oppsettet av rapporten. Rapporten er delt opp i hovedkapitler med underkapitler som hører til hovedkapitlene. For siteringer og refereanser av eksterne kilder har gruppen valgt å bruke vancouver-stil, det vil si at referansene er punktummerert opp mot en referanseliste i slutten av rapporten. Alle bilder, tabeller og lister er representert i egne innholdsfortegnelser i begynnelsen av rapporten.

2 Kravspesifikasjon

2.1 Dagens applikasjon

For å kunne vite hva som skal inn i den nye applikasjonen og vite hva som funker og ikke funker med dagens applikasjon så fikk vi en lang og god fremvisning av både den fulle nettsiden og av dagens mobilapplikasjon på det første møtet med EVRY. Dagens mobilapplikasjon er egentlig bare en mobilversjon av nettsiden med begrenset funksjonalitet, slik at det var denne gruppen fikk vist.

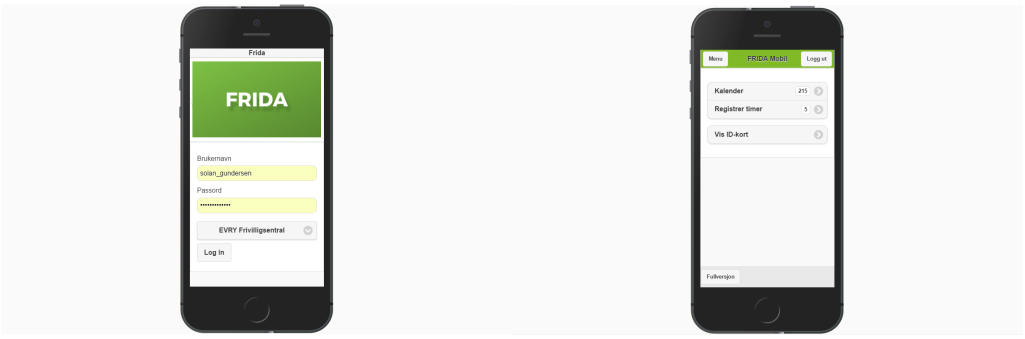
Problemet med dagens mobilapplikasjon er som sagt at dagens app ikke har så mye funksjonalitet i forhold til nettsiden og hva som er ønskelig. I tillegg så er dagens app ikke like brukervennlig som den kunne ha vært. Derfor ble gruppen vår bedt om å lage en ny app hovedsaklig for frivillige, men der gruppen kunne utvide til flere roller om det ble nok tid.

Som vi kan se av figur 1, 2, 3 og 4 så er ikke alt like praktisk eller optimalt anlagt. Dette ser vi spesielt i figur 2, der det man kan gjøre i appen, er listet opp på en lite brukervennlig måte da man kun bruker tekst og små knapper. I tillegg kan vi se på figur 4 at det ikke er en veldig lur og sikker måte å vise Id-kort på. Hvordan disse delene har blitt forbedret i den nye appen forklarer vi senere i rapporten i del 5.3.5.

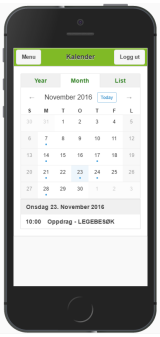
2.2 Operasjonelle krav

Det ble ikke gitt noen spesifikke operasjonelle krav untatt at appen skulle fungere med det eksisterende FRIDA-systemet. Derfor kom gruppen opp med noen flere krav for å ha noe å forholde oss til. Totalt sett ble de operasjonelle kravene derfor disse:

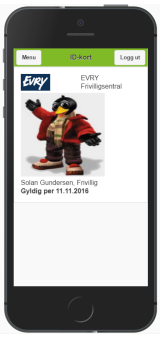
- Appen skal funke med det eksisterende FRIDA-systemet.
- Appen skal ha en oppetid på rundt 99%.
- Appen skal kunne laste informasjon innen et par sekunder.
- Appen skal funke på både iOS og Android.
- Appen skal være enkel å vedlikeholde.



Figur 1: Dette viser logg inn siden til dagens app. [5] Figur 2: Dette viser menyen til dagens app. [5]



Figur 3: Dette viser kalenderen som eksisterer i dagens app. [5]



Figur 4: Dette viser visningen av ID-kort slik det vises i dagens app. [5]

I del 5.1 blir statusen til de operasjonelle kravene i appen nærmere forklart.

2.3 Sikkerhetskrav

Sikkerhet er en viktig del av etthvert prosjekt. Da dette prosjektet inneholder en del sensitiv informasjon er dette spesielt viktig. EVRY ga oss ikke noen spesifikke krav, men at vi skulle legge på all den sikkerhet vi følte var nødvendig. Under er det listet opp noen sikkerhetskrav vi følte var nødvendige før det settes i produksjon.

- Api og app skal kun vise informasjon som en innlogget bruker kan se. Vanlige frivillige kan altså kun se informasjon om seg selv, om tjenester i sin egen organisasjon som de har lov til å se, etc. De kan derimot ikke se masse informasjon om andre brukere. Administratorer kan derimot også se informasjon om andre frivillige i sin organisasjon.
- Database skal ikke kunne bli tatt med ”SQL-injection”.
- Databaseinnhold bør være sanitert for html-tags.
- Innlogget bruker må logges ut (gå til innloggingsside) med en gang informasjonen som lagres om bruker forsvinner (sjekk ved hver side).

I del 4.3 blir statusen til sikkerhetskravene nærmere forklart. I tillegg vil det stå mer om sikkerhet.

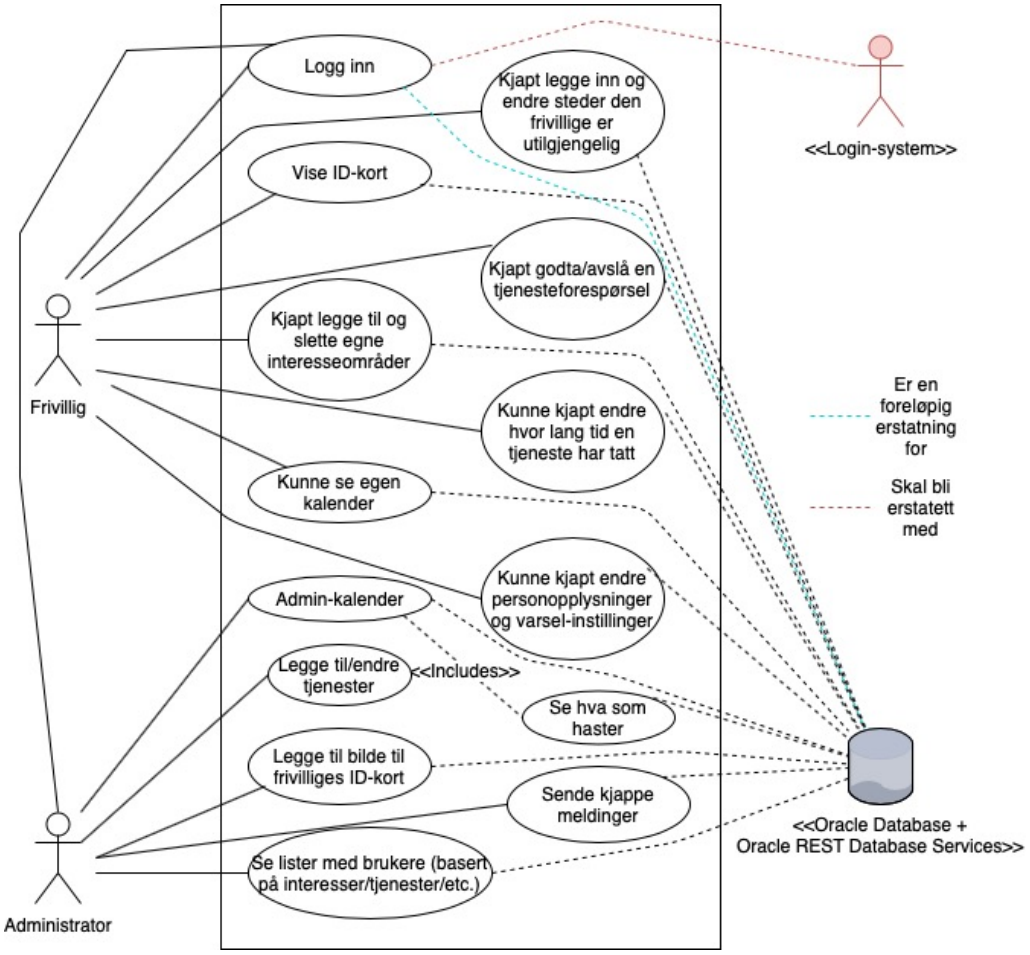
2.4 Krav til grensesnitt

2.4.1 Krav til brukergrensesnitt

EVRY hadde ikke noen spesifikke krav til grensesnitt. Men siden FRIDA brukes av blant annet frivillige som er litt eldre, så er det viktig at spesielt frivillig-delen av appen er så brukervennlig at alle brukergrupper kan bruke den. Det samme vil gjelde for en evt. tjenestemottaker-del om dette blir publisert en gang. En administrator vil derimot ikke nødvendigvis trenge like mye brukervennlighet for alle grupper da disse normalt sett er litt mer oppegående. Les mer om hvordan designet av brukergrensesnittet er løst i del 3.3.

2.4.2 Krav til kommunikasjon mellom forskjellige grensesnitt

Som nevnt i del 2.2, så må systemet kunne kommunisere med resten av FRIDA-systemet. Om appen ikke er koblet til systemet er det ikke vits i å ha appen da appen skal være et tillegg til



Figur 5: Den fullstendige use-case diagrammet med admin

appen.

I tillegg til å være koblet til databasen må appen i produksjon være tilkoblet innloggingssystemet. Dette er et separat system som ikke er koblet opp mot databasen. Dette systemet har EVRY derimot sagt at vi ikke trenger å koble oss opp mot da de mener vi ikke trenger å lære oss dette systemet for å lage innloggingen.

2.5 Use-case diagram

For å kunne se de forskjellige tingene de forskjellige brukergruppene skal kunne gjøre i appen har gruppen satt opp et use-case diagram. I figur 5, så kan man se både det den frivillige og administrator skal kunne gjøre. Man kan også se at det eksisterer et eget login-system som den nye appen vil koble seg opp mot i produksjon, men, som det også er nevnt i del 2.4.2, at oppgaven ikke omfatter å koble appen opp til dette systemet og at det derfor har blitt laget et midlertidig login-system opp mot databasen i utviklingsprosessen. Vi kan også se at all kommunikasjon er, untatt login når dette er satt opp, med databasen via REST-tjenesten ”Oracle REST Database Service” (ORDS).

Det kan utifra use case-diagrammet også bli sett at administrator kan gjøre mye admini-

strering, men appen skal ha den begrensning at det skal være enkle administratoroppgaver, slik at hvis man skal opprette et stort arrangement med mange innstillinger, så er det forbeholdt at det skal opprettes på en datamaskin istedenfor i appen.

2.6 Brukerhistorier

Brukerhistorier kan brukes til å enkelt fortelle og se hva som skal kunne gjøres i et program. Under listes opp en rekke brukerhistorier som forteller hva appen skal gjøre. Vi bruker brukerhistorier fordi vi da kan høre med de forskjellige brukergruppene hva de ønsker å ha i en FRIDA app. Dette gjør det enklere å se hvilke funksjoner som er ønsket i appen fra brukernes synspunkt.

2.6.1 Brukerhistorier rundt profil og ID-kort

- Som en frivillig som nettopp har byttet jobb må jeg ha mulighet til å endre epost til min nye jobb-e-post da det er denne e-posten jeg bruker.
- Som en frivillig som nettopp har giftet meg må jeg oppdatere etternavnet mitt da dette er endret.
- Som en frivillig må jeg kunne bevise at jeg virkelig jobber som frivillig slik at brukeren vet at jeg ikke er en bedrager da dette har skjedd.
- Som en administrator så må jeg enkelt kunne ta bilde av en ny frivillig til deres ID-kort eller av en gammel frivillig fordi de har forandret utseendet.

2.6.2 Brukerhistorier rundt kalender

- Som en frivillig som ønsker å sjekke hvordan timeplanen er i nærmeste fremtid slik at jeg kan vite når jeg har ledig tid eller ikke.
- Som en frivillig som ønsker å se hva jeg gjorde forrige måned slik at jeg kan gjøre regnskap om nødvendig.
- Som en frivillig som ønsker å se hva som vil skje av ting rundt mine interesser den nærmeste tiden da jeg gjerne vil være med på noe eller utføre et arbeid den nærmeste tiden.
- Som en frivillig som ønsker å se hva som vil skje av ting i min frivillighetssentral den nærmeste tiden. Dette da jeg har lyst til å delta på noe snart.
- Som en administrator som ønsker å se hvilke tjenester det haster å gjøre noe med (få nok frivillige på). Dette slik at det er mulig å gjennomføre disse tjenestene.
- Som en administrator som ønsker å se alle tjenester i min frivillighetssentral for å enklere kunne administrere disse.
- Som enten frivillig eller administrator ha mulighet til å gå direkte fra kalender og over til en detaljside for en tjeneste for å enkelt sjekke mer rundt denne tjenesten.

2.6.3 Brukerhistorier rundt interesser

- Som en frivillig å kunne se hvilke interesser jeg har slik at jeg evt. kan oppdatere disse.
- Som en frivillig å enkelt kunne legge til flere interesser om jeg har fått meg dette.
- Som en frivillig å enkelt kunne fjerne interesser om jeg har mistet interessen for et interesseområde.

2.6.4 Brukerhistorier rundt tjenester

- Som en frivillig ha mulighet til å se tjenester jeg har satt meg opp på slik at jeg vet når og hvor dette er.

- Som en frivillig ha mulighet til å se interessante tjenester og melde meg på dem enkelt.
- Som en frivillig å enkelt kunne se og godta/avslå en tjenesteforspørsel.
- Som en administrator å enkelte opprette nye tjenester fra appen.
- Som en administrator å enkelt kunne redigere tjenester fra appen.

2.6.5 Brukerhistorier rundt utilgjengelighet

- Som en frivillig å enkelt kunne fortelle at jeg er på trening hver mandag fra 20.00-21.00 slik at jeg ikke blir spurt om å ta oppdrag i dette tidsrommet.
- Som en frivillig å enkelt kunne fortelle når jeg er på ferie slik at jeg ikke blir spurt om å ta oppdrag i dette tidsrommet.
- Som en administrator å kunne se hvem som er ledig og ikke ledig på et gitt tidspunkt i tilfelle det blir hasteoppdrag.

2.6.6 Brukerhistorier rundt det å endre timer på en tjeneste

- Som en frivillig å enkelt kunne fortelle at en tjeneste tok lenger tid enn planlagt.
- Som en frivillig å enkelt kunne fortelle hvorfor en tjeneste tok lenger tid enn planlagt.

2.6.7 Brukerhistorier rundt lister

- Som en administrator å enkelt kunne finne frivillige personer i min egen organisasjon som er interessert i å gjøre et oppdrag.
- Som en administrator kunne få kontaktinformasjon til en spesifikk frivillig.
- Som en administrator kunne se hvilke personer som har vært frivillige i en spesifikk tjeneste.

2.7 Essensielle funksjonaliteter som ble planlagt skulle inn i appen

Basert på use-case diagrammet i figur 5 og brukerhistoriene nevnt i 2.6, så har vi kommet opp med disse essensielle funksjonalitetene. Disse essensielle funksjonalitetene er listet opp under med en begrunnelse på hvorfor de er essensielle for henholdsvis frivillige og admin.

2.7.1 For frivillige

Siden det ble planlagt at frivillige skulle prioriteres først, ble disse essensielle funksjonene planlagt tidlig i prosessen.

Vise sikkert, digitalt, ID-kort

For å kunne sikkert identifisere seg selv så må frivillige kunne ha et ID-kort som kan bekrefte hvem de er. Dette ID-kortet må være laget slik at man kan se at det er ekte og ikke bare et manipulert bilde.

Kunne se og bli med på tjenester

Er det noe som raskt bør kunne gjøres er det å finne og bli med på tjenester. Dette for at en frivillig skal kunne utføre arbeid og fordi man enkelt skal kunne ta hasteoppdrag.

Kunne se og oppdatere interesser

Frivillige kan bli kontaktet av administratorer om å ta et oppdrag. Derfor er det viktig å ha oppdaterte interesseområder slik at administratorene vet om en frivillig kan være interessert i dette oppdraget.

Kunne se og endre profil

For en frivillig er det lurt at man kan se og oppdatere profilen med kontaktinformasjon, navn, etc. Dette fordi man derfor slipper å måtte gå inn på nettsiden for å oppdatere ett felt.

Kunne oppdatere timer fort i appen

Selv om administratorene som setter opp tjenestene prøver å sette opp korrekte antall timer tjenesten varer, kan dette i noen tilfeller bli forlenget eller forkortet. For å kjapt kunne endre dette tidspunktet til korrekt tid mens man går fra tjenesten slik at man slipper å måtte notere tiden man var ferdig for å senere å måtte gå på en datamaskin for å oppdatere tidspunktet, er dette viktig å ha med.

Kunne oppdatere tidspunkter på utilgjengelighet

Som nevnt ang. interesser over, så kan administratorer ta kontakt med spørsmål om å ta et oppdrag. Men istedenfor å kun se på interessene til den frivillige så kan en admin også se på om en frivillig er tilgjengelig eller ikke. For at det skal være kjapt og enkelt å oppdatere disse tidene, enten det er ukentlige hendelser som faste treninger, eller ferier, bør dette være i appen.

Kunne vise kalender med frivilliges tjenester lagt inn

For at en frivillig enkelt skal se hvilke tjenester som evt. er på en dag så vil det være lurt å vise dette i en kalender.

2.7.2 For admin

Siden appen for administratorer ble påbegynt senere i programmeringsprosessen så ble disse essensielle funksjonene planlagt litt senere i prosessen.

Kalender over alt som skal skje i organisasjonen

Som administrator er det viktig å ha enkel oversikt over alt som skjer i organisasjonen. Derfor ønskes det at alle tjenester vises i en kalender.

Oversikt over de nærmeste tjenestene som mangler nok frivillige

Det ønskes et sted man kjapt kan se hvilke tjenester i den nærmeste tiden som mangler nok frivillige. Det ønskes at tjenestene som mangler frivillige for de 10 neste dagene vises i denne listen.

Lage nye tjenester fra appen

Det er sterkt ønskelig at det skal være mulig for en administrator å legge inn en ny tjeneste mens de er på farten. Tjenestene trenger ikke å ha mange innstillinger da store events mest sannsynlig vil bli planlagt på en datamaskin. Men det er ønskelig at enkle tjenester som å handle mat for en tjenestemottaker kan opprettes fra appen.

Oversikt over brukere og deres tjenester, interesser og lignende

Det er ønskelig at en administrator kan se hvilke brukere som er på en tjeneste og alle brukere i deres organisasjon. Det er også ønskelig at deres profil med deres tjenester, interesser og lignende for å både kunne se deres tjenester og finne den perfekte frivillige for en tjeneste.

2.8 Mindre viktige funksjonaliteter

Basert på use-case diagrammet i figur 5 og brukerhistoriene nevnt i 2.6, så har vi kommet opp med disse mindre viktige funksjonalitetene. Disse mindre viktige funksjonalitetene er listet opp under med en begrunnelse på hvorfor de er mindre viktige men alikevel lure og ha med for henholdsvis frivillige og admin.

2.8.1 For frivillige

Disse funksjonene er mindre viktige, men allikevel ønsket til bruk for frivillige.

Filtrere tjenester i kalender

Det kan være aktuelt å kunne vise mer enn kun det en frivillig har meldt seg på i kalenderen. Dette vil i så fall bli lagt til som en filtreringsknapp.

Varsling

Det er ønskelig at det blir integrert notifikasjonsvarslinger for de som har appen. Med dette menes varslinger slik som når en frivillig blir spurt om å ta et oppdrag så kan den frivillige bli tatt direkte til tjenesten for å si ja eller nei. I tillegg kan det brukes til å minne frivillige på at en tjeneste begynner snart.

Selv om dette er ønskelig er dette en stor oppgave som ikke er førsteprioritet da funksjonalitet er mer viktig enn nye varslingstjenester.

2.8.2 For admin

Disse funksjonene er mindre viktige, men allikevel ønsket til bruk for administratorer.

SMS og notifikasjonsutsendelse til frivillige

Det er ønskelig, men mindre viktig, å kunne sende ut SMS og notifikasjonsvarsler fra appen for administratorer. Dette inkluderer både notifikasjoner om at det er ønskelig at en frivillig tar et oppdrag, som lest mer om over, og at det sendes SMS'er via en meldingsutsendelsesside på admin sitt kontrollpanel.

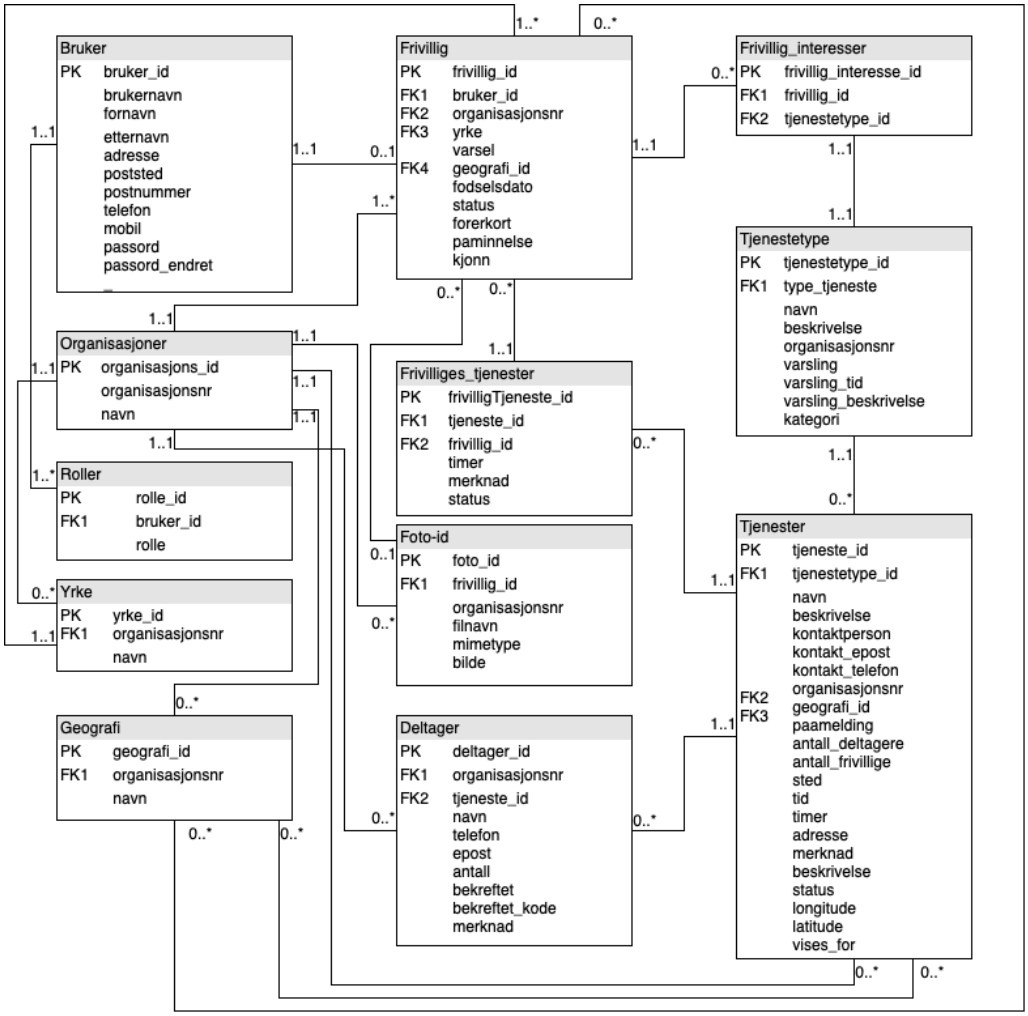
Meldingsutsendelse, enten det er masseutsending til en gruppe, eller til bare en person, er ment for å kjapt sende meldinger til for eksempel de som skal være frivillige på en tjeneste. Det kan også være personlige meldinger. Dette har ikke stor prioritert da administratorer mest sannsynlig kan sitte ved en PC når dette trengs, selv om det kan skje noe som haster.

Ta bilde av frivillige for ID-kort

Det er veldig ønskelig å forenkle opplastning av bilde av frivillige, slik at man istedenfor å måtte legge bilde inn på PC'en for å laste opp eller gå inn på fullversjonen av nettsiden på mobil, å kunne gjøre dette direkte i appen. Men selv om dette er veldig ønskelig er ikke dette blant det viktigste å ha tilgjengelig for administrator på mobil.

2.9 Forenklet klassediagram for databasen

Vi har laget et forenklet klassediagram for databasen som man kan se i figur 6. Med forenklet menes at det bare har blitt tatt med de tabellene og kolonnene som er relevante eller interressante for appen som har blitt laget. I tillegg, for å beholde en viss sikkerhet rundt databasen så har man brukt andre navn i klassediagrammet enn hva som nødvendigvis er navnet i databasen.



Figur 6: Et forenklet klassediagram for databasen.

2.9.1 Generell beskrivelse

Som man kan se fra klassediagrammet så er brukertabellen i databasen en samlet tabell for alle brukere, enten de er frivillige, administratorer eller tjenesteytere. Deretter fins det en frivilligtabell med utfyllende informasjon som omhandler en frivillig. Det eksisterer også en tjenesteyter-tabell med informasjon om disse, men administratorer har ingen egen tabell. Denne delen av databasen er derfor ganske normalisert.

Deler som ikke er normalisert er blant annet at man bruker ”organisasjonsnr” som referanse istedenfor ”organisasjons_id”. Dette gjør at, selv om organisasjonsnummeret vanligvis ikke endres, at hvis noe skulle skje som gjør at dette skulle endre seg, så får man ikke endret dette på en enkel måte. Et eksempel kan være at en organisasjon som bruker FRIDA slår seg sammen med en organisasjon som ikke bruker det enda, og den organisasjonen som ikke har brukt det enda blir organisasjonens nye navn og nummer. I tilfelle så er det muligens bedre å viderebruke kontoen den første organisasjonen bruker fra før av enn å starte en helt ny tjeneste. Men dette blir som sagt ikke like enkelt å gjøre.

En frivillig kan registrere flere interesser. Disse interessene er forskjellige tjenestetyper som er brukt i registreringen av tjenester. Tjenester blir registrert ved å legge til en tjeneste til tjeneste-tabellen, mens for hver frivillig som melder seg på tjenesten blir det opprettet en ny rad i ”Frivilliges_tjenester” der en tjeneste er koblet sammen med den frivillige.

2.9.2 Forklaringer for kolonnene i tabellene

Merk at selv om det ikke spesifikt står at en kolonne ikke trenger å oppgis betyr ikke det nødvendigvis at man må oppgi denne. Når vi spesifikt har skrevet at man ikke trenger å oppgi dette er det for å spesifisere dette da man kan tro at dette er nødvendig.

Brukertabellen

Tabellen ”Bruker” har blant annet disse kolonnene:

- bruker_id: Id for brukere.
- brukernavn: Brukernavnet man bruker for å logge inn med.
- fornavn: Brukerens fornavn.
- etternavn: Brukerens etternavn.
- adresse: Brukerens adresse.
- poststed: Brukerens poststed.
- postnummer: Brukerens postnummer.
- telefon: Telefonnummeret til brukeren, må ikke oppgis.
- mobil: Mobilnummeret til brukeren, må ikke oppgis.
- passord: Passordet til testbrukeren (uten beskyttelse). Dette var ikke i databasen når vi fikk den, men ble lagt til da de bruker et annet passordsystem. Les mer om dette senere i rapporten.
- passord_endret: sier om testpersonens passord har blitt bedt om å bli endret.

Frivilligtabellen

Tabellen ”Frivillig” har blant annet disse kolonnene:

- frivillig_id: Id til den frivillige.
- bruker_id: Brukerens id. Dette er fra ”Bruker”-tabellen.
- organisasjonsnr: Organisasjonsnummeret til frivilligorganisasjonen den frivillige er med-

lem av.
- yrke: Den frivilliges yrkesstatus.
- varsel: Hvordan den frivillige skal bli varslet på (SMS/ringes/etc.).
- geografi_id: Id'en til et sted angitt i "Geografi"-tabellen.
- fodselsdato: Fødselsdatoen til den frivillige. Det eksisterer også dødsdato selv om dette ikke er tatt med diagrammet.
- status: Er den frivillige aktiv, inaktiv eller noe annet i organisasjonen.
- forerkort: Har den frivillige førerkort? Det kan også bli registrert type i et annet felt.
- paaminnelse: Hvordan den frivillige skal bli kontaktet i forhold til påminnelser.
- Kjonn: Hvilket kjønn er den frivillige.

Frivillig_interesser-tabellen

Tabellen "Frivillig_interesser" har blant annet disse kolonnene:

- frivillig_interesse_id: Id for frivilliginteressen.
- frivillig_id: Den frivilliges id.
- tjenestetype_id: Tjenestetype-id'en. Tjenestetype er det som brukes som "interesse".

Tjenestetype-tabellen

Tabellen "Tjenestetype" inneholder forskjellige "tjenestekategorier", selv om det også eksisterer en "tjenestekategori"-tabell med mer generelle kategorier som "Aktivitet". Tjenestetype-tabellen inneholder derimot ting som "Leksehjelp". Tabellen inneholder blant annet disse kolonnene:

- tjenestetype_id: Tjenestetypens id.
- type_tjeneste: Id'en til "tjenestekategori"-tabellen. Denne tabellen har vi ikke tatt med da den ikke er brukt av oss (gruppen har brukt kolonnen kategori, som egentlig forteller det samme, bare at det er i tekst direkte).
- navn: Navnet/Tittelen på tjenestetypen.
- beskrivelse: En beskrivelse av tjenestetypen.
- organisasjonsnr: Organisasjonsnummeret til frivilligorganisasjonen som bruker denne.
- varsling: Er det varsling satt opp på denne typen?
- varsling_tid: Hvor lang tid i forveien skal det varsles?
- varsling_beskrivelse: Hva skal det stå i varslingen.
- kategori: En generell kategori gitt i tekstform, er mye av det samme som det som er i "Tjenestekategori"-tabellen.

Tjenester-tabellen

Tabellen "Tjenester" inneholder rader med en spesifikk tjeneste. Med det menes en spesifikk dato. Tabellen inneholder blant annet disse kolonnene:

- tjeneste_id: Id for denne tjenesten.
- tjenestetype_id: Id for denne tjenestens tjenestetype. Denne id'en kan for eksempel referere til tjenestetypen "Leksehjelp", og da er i tilfelle denne tjenesten en dag og tid dette er arrangert (av flere ganger mest sannsynlig).
- navn: Navnet/Tittelen på denne tjenesten (for eksempel "Leksehjelp på Flåklype ungdomsskole").
- beskrivelse: En nærmere beskrivelse for denne tjenesten.

- kontaktperson: Navnet til kontaktpersonen. Dette må ikke oppgis.
- kontakt_epost: Epost til kontaktpersonen. Dette må ikke oppgis.
- kontakt_telefon: Telefonen til kontaktperson. Dette må ikke oppgis.
- organisasjonsnr: Organisasjonsnummeret til frivilligorganisasjonen som har tjenesten.
- geografi_id: Dette er en av flere måter å oppgi stedet det skal arrangeres.
- paamelding: Er det nødvendig å melde seg på på forhånd?
- antall_deltagere: Om påmelding er slått på, er maks antall deltagere oppgitt her.
- antall_frivillige: Ønsket antall frivillige.
- sted: Dette er en av flere måter og oppgi sted på.
- tid: Dato og tid for starttidspunktet for tjenesten.
- timer: Antall timer det skal vare (oppgitt i desimaler, slik at 1,5 er en og en halv time).
- adresse: Dette er en måte og oppgi sted på.
- merknad: Dette er en av to beskrivelser/merknader ang. tjenester.
- beskrivelse: Dette er en av to beskrivelser/merknader ang. tjenester.
- status: Er tjenesten aktiv, kansellert, eller lignende.
- longitude: Dette er en av flere måter å oppgi stedet det skal arrangeres.
- latitude: Dette er en av flere måter å oppgi stedet det skal arrangeres.
- vises_for: Hvem skal kunne se tjenesten (alle, offentlig, frivillige, etc.).

Frivilliges_tjenester-tabellen

Tabellen ”Frivilliges_tjenester” inneholder rader der en rad kobler sammen en frivillig til en tjeneste, slik at en tjeneste har like mange rader som det er påmeldte frivillige. Tabellen inneholder blant annet disse kolonnene:

- frivilligTjeneste_id: Id for koblingen mellom den frivillige og tjenesten.
- tjeneste_id: Id’en til tjenesten den frivillige er koblet til.
- frivillig_id: Id’en til den frivillige som er koblet til tjenesten.
- timer: Antall timer egentlig brukt. Default er det samme som tjenestens timer, men kan bli endret på individuelt. Dette er også oppgitt i desimaler.
- merknad: Kan brukes til å forklare evt. endringer i timer (som for eksempel ”Måtte være med å vaske opp, dette tok lenger tid enn forventet”.
- status: Er deltagelsen som frivillig bekreftet?

Deltager-tabellen

Deltager-tabellen inneholder deltagere som har meldt seg på en tjeneste, hvis det er påmeldingskrav. Altså hvis kolonnen ”paamelding” er satt til ”Ja”. Tabellen inneholder blant annet disse kolonnene:

- deltager_id: Id for raden i denne tabellen.
- organisasjonsnr: Organiusasjonsnummeret til organisasjonen til tjenesten som er referert.
- tjeneste_id: Id til tjenesten deltageren har meldt seg opp på.
- navn: Navnet på personen som registrerer deltagelsen.
- telefon: Telefonnummeret til personen som registrerer deltagelsen. Brukes til å blant annet bekrefte dette nummeret ved å sende en SMS med bekreftelseskode.
- epost: Epost til denne deltageren.
- antall: Antall påmeldte som blir påmeldt denne runden.

- bekreftet: Er telefonnummeret bekreftet.
- bekreftet_kode: Koden som sendes ut for å bekrefte.
- merknad: En merknad ang. deltagelsen.

Foto-id-tabellen

Tabellen ”Foto-id” lagrer bilder brukt til id-kort. Tabellen inneholder blant annet disse kolonnene:

- foto_id: Id for hver rad i denne tabellen.
- frivillig_id: Id for den frivillige det er bilde av.
- organisasjonsnr: Organisasjonsnummeret til den organisasjonen den frivillige er med på.
- filnavn: Filnavnet til filen som ble opplastet.
- mimetype: Mimetypen til filen (som jpeg, png, etc.).
- bilde: En blob med inneholdet for å lage bildet.

Organisasjoner-tabellen

Tabellen ”Organisasjoner” inneholder informasjon om frivillige organisasjoner i systemet. Tabellen inneholder blant annet disse kolonnene:

- organisasjons_id: Id til denne organisasjonen i systemet. Brukes ikke da organisasjonsnummeret brukes i stedet.
- organisasjonsnr: Organisasjonsnummeret til organisasjonen. Brukes som ”foreign key” i andre tabeller.
- navn: Navnet på organisasjonen.

Roller-tabellen

Tabellen ”Roller” inneholder hvilken rolle brukeren har i systemet. En bruker kan ha flere roller. Tabellen inneholder blant annet disse kolonnene:

- rolle_id: Id for raden i denne tabellen.
- bruker_id: Id for brukeren.
- rolle: Et tall som identifiserer rollen. Disse rollene kan være tjenesteyter, frivillig, administrator eller lignende.

Yrke-tabellen

Tabellen ”Yrke” inneholder forskjellige yrker eller yrkesstatuser. Tabellen har blant annet disse kolonnene:

- yrke_id: Id for raden i denne tabellen.
- organisasjonsnr: Organisasjonsnummeret til organisasjonen som opprettet yrket/yrkesstatusen.
- navn: Navnet på yrket/yrkesstatusen.

Geografi-tabellen

Tabellen ”Geografi” inneholder forskjellige steder som kan bli referert til via ”geografi_id”-kolonnene i andre tabeller. Tabellen inneholder blant annet disse kolonnene:

- geografi_id: Intern id til dette stedet.
- organisasjonsnr: Organisasjonsnummeret til organisasjonen som opprettet stedet.

- navn: Navnet på stedet.

3 Programvarearkitektur og design

3.1 Utviklingsverktøy

I alle arbeid er det viktig å stille med riktige og gode verktøy, dette gjelder også i programvareutvikling. For dette prosjektet har gruppen vår valgt å bruke en kombinasjon av verktøy som er både nye og gamle i forhold til vår erfaring.

3.1.1 Valg av programmeringsspråk

Den første og viktigste avgjørelsen gruppen måtte ta før utviklingen kunne begynne var hvilket utviklingsrammeverk som skulle brukes. Tidligere har medlemmene av gruppen benyttet seg av Android Studio [6] og skrevet mobilapplikasjoner for Android gjennom Java og XML.

Når gruppen vår skulle velge hvilket programmeringsspråk og rammeverk som skulle brukes så hadde vi flere språk å velge mellom. Siden vi skulle lage en ny app, så kunne vi velge å lage en app for Android og en for iOS. Men det eksisterer også mange rammeverk som lar en skrive en app i ett språk, også blir den enten konvertert til en native app for hver platform den blir laget for, eller kjørt via et WebView, med native kode for hver plattform gitt av rammeverket selv. Det eksisterer flere slike rammeverk. Noen eksempler er Ionic, React Native, Xamarin, PhoneGap, Flutter, Corona SDK, JQuery Mobile, Oracle APEX og Oracle JET.

De største kravene vi hadde var at det skulle være enkelt å bruke, slik at vi kunne bruke tiden på å skrive funksjonalitet istedenfor å lære rammeverket, og at vi ikke trenger å skrive kode flere ganger for flere platformer. Det skal også være mulig for EVRY å enkelt kunne endre kode hvis de velger å bruke vår app.

Native app

Hvis vi ser på funksjonalitet, så ville nok det beste alternativet være å lage native applikasjoner. Men vi bestemte at appen ikke skulle lages slik fordi vi fant ut at hvis dette ble valgt, så ville en stor del av tiden gå til å kode for to apper, noe som ville gjøre at vi fikk mindre tid til å jobbe på flere funksjonaliteter. I tillegg så hadde vi bare et gruppemedlem med en Mac, så hvis vi skulle lage en native iOS-app, så kunne kun en av oss jobbe med denne appen. Gruppen kunne også bare velge å lage en app for Android, siden vår bacheloroppgave ikke spesifiserte å støtte flere platformer, men som nevnt i 2.2, så bestemte gruppen at vi skulle støtte begge platformer. Dette ble bestemt da det ville bli enklere for EVRY å bruke appen i produksjon hvis den også virket for iOS.

React Native

React Native er et rammeverk som bruker Javascript og React for å lage native apper. Mens de fleste andre rammeverk nevnt under bruker WebView for å vise apper (”hybrid apps”), så konverterer React Native koden til native apper. Grunnen til dette er at de bruker ”the same fundamental UI building blocks as regular iOS and Android apps” [7].

React Native har en stor brukerbase på nett. Dette betyr at man mest sannsynlig kan finne svaret man så etter hvis man søker på internett. Dette var viktig da EVRY ikke hadde mye erfaring med å bruke React, så ved å velge dette språket måtte vi støtte oss hovedsaklig på

internett-støtte. React Native støtter også funksjoner som kart og andre komponenter, hovedsaklig laget av tredjeparter eller deg selv med native kode. For eksempel er kartkomponenten som blir brukt mest laget av Airbnb.

Det er to måter å lage apper med React Native: Du kan enten bruke et rammeverk i React Native som heter Expo, som er et verktøy som gjør det mulig å lage apper med plugins/biblioteker. Dette virker så lenge all funksjonalitet er tilgjengelig i de bibliotekene som støttes av Expo. Dette betyr at du ikke kan legge til dine egne biblioteker med native-kode. Det andre alternativet er å bruke den originale React Native, men i dette tilfellet kreves det at man har en Mac hvis man ønsker å jobbe med iOS-kode. Dette trengs ikke i Expo fordi de har tilgang til Mac'er online for konvertering.

Oracle APEX

Den originale FRIDA-appen og nettsiden er laget i Oracle APEX. Dette er et rammeverk der HTML, Javascript og CSS er brukt sammen med ferdige komponenter og "pek og klikk" visual builder, altså en visuell måte å enkelt lage apper på.

Selv om dette ville være en enkel måte å lage appen på, så tenkte gruppen vår at dette ikke ville passe for vår bacheloroppgave da det ser ut til å være veldig basert på visuell programmering. Det ville også gi oss restriksjoner da den har begrenset mulighet til å manipulere appen. I tillegg til at vi ikke er vant med å bruke denne formen for koding.

Oracle JET

Oracle JET er et annet, nyere rammeverk for å lage nettsider og hybride mobilapper. Som med Oracle APEX, så har den en pek og klikk-editor på nett kalt "Visual builder". Men den har i tillegg et kommandolinjeprogram som lager og kompilerer apper som du kan lage med Javascript, HTML og CSS. Dette betyr at det er enklere å lage mer fleksible apper som en programmerer.

Oracle JET er ikke egentlig sitt eget rammeverk, men heller et rammeverk av rammeverk. Dette fordi Oracle JET bruker forskjellige tredjepartsbiblioteker for å løse problemer. Dette kan både være bra og dårlig, da du enkelt kan implementere andre biblioteker selv, men det kan være problemer med noen problemer som man ikke kan fikse selv. For å kompilere hybride apper så brukes biblioteket Cordova, som du kan lese mer om i del 3.1.2.

Oracle JET inneholder også egne komponenter som lister, avatar- og tekst-felt som kommer ferdigstyled for å se ut som Android/iOS-komponenter om mulig. Oracle JET har ikke en like stor brukerbase på nett som React Native har, men de har imidlertid deres egen kokebok som forklarer hvordan man bruker de fleste komponenter. I tillegg har Oracle blogg-poster med god informasjon om mer kompleks funksjonalitet. Men en ulempe er at, for å få noe informasjon i det hele tatt, må man vite hvilket bibliotek eller plugin hver del av rammeverket er hentet fra for å finne mer informasjon om det på nett. For eksempel, for å få informasjon om observable-variabler, så må man skrive "knockout observable variable" istedenfor "oracle jet observable variable" når man søker på Google, siden observable-variabler er hentet fra Knockout-biblioteket.

Andre rammeverk

Andre rammeverk slik som Ionic, PhoneGap og Xamarin var kjapt diskutert blant gruppemedlemmene. Mens Ionic og PhoneGap også bruker Cordova som en måte å kompilere hybridappen fra HTML, CSS og Javascript, så bruker Xamarin C# til å lage apper i Visual Studio

på Windows eller Xamarin studio på Mac. For å lage en iOS app på Windows med Xamarin så trengs desverre fremdeles en Mac koblet trådløst til Windows-maskinen for å kompilere kode fortløpende. Men Xamarin har en visuell designer lik Android Studio og XCode har, slik at det er mulig å lage apper og kode uten å måtte definere alle designelementer i kode.

Sammenligning av rammeverk og løsninger

I tabell 1 så kan man se en fin sammenligning av de forskjellige språkene og rammeverkene vi vurderte. Som tidligere nevnt så bestemte gruppen seg for å ikke kode en native app siden vi da måtte ha laget to separate apper. Da betydde det at vi hadde seks rammeverk å velge mellom. Selv om PhoneGap og Ionic ligner Oracle JET, så har de heller ikke en stor brukerbase, og vi kunne heller ikke fått hjelp av EVRY hvis vi skulle velge en av disse rammeverkene. Xamarin har også en mindre brukerbase og bruker C#, noe som gruppen ikke var like kjent med som gruppen var med for eksempel Javascript. Oracle APEX var også kjapt diskutert siden den originale FRIDA-appen er laget med det. Men gruppen bestemte at det ikke ville være en god ide da det ikke ville være en ekte hybrid app, og siden mye av koden er generert når appen kjører, noe som gjør den vanskeligere å redigere.

Da hadde vi React Native og Oracle JET å velge mellom. Vi kunne noe React Native fra før av, mens Oracle JET var noe EVRY foreslo, siden de jobber med Oracle. Etter å ha sett mange videoer og informasjon om React Native og Oracle JET, bestemte vi oss for å gå for Oracle JET. Måten gruppen tenkte da vi valgte dette var at, selv om React Native har en større brukerbase, så har Oracle JET en stor kokebok med mange eksempler for hvordan komponentene brukes. I tillegg hadde EVRY folk som kunne hjelpe til med Oracle JET. En annen grunn til at vi valgte Oracle JET var at EVRY bruker en Oracle database, som man kan lese mer om i del 3.1.5. På dette tidspunktet trodde vi nemlig at Oracle JET muligens hadde en form for snarvei for å kontakte databasen. Det viste seg siden å ikke stemme, men at man måtte bruke tjenesten ORDS istedenfor. Denne tjenesten kan lese mer om i del 3.1.6. For en lengre diskusjon rundt dette, les del 8.1

3.1.2 Om Cordova

Cordova er en tjeneste som lager apper for flere platformer fra en kodebase. Cordova er gratis og open source. Cordova bruker WebView, eller WKWebView på iOS, for å kjøre appen. For å kommunisere med telefonens hardware og native programvare, slik som å ta bilder eller få sensor-informasjon, så brukes plugins, enten tredjepartsplugins eller plugins laget av Cordova-teamet selv.

3.1.3 Git og Bitbucket

For å kunne dele koden med hverandre fortløpende brukte gruppen versjonskontrollsystemet Git. Som tjenesteleverandør for Git ble Bitbucket valgt da NTNU har avtale med dem og fordi de har koblinger til tjenester som Jira og Trello.

3.1.4 Trello

Trello er et verktøy som lar brukere opprette en tavle, på tavlen setter man opp ulike kategorier som man kan fylle med Trello-kort. Trello-kortene kan tildeles medlemmer og flyttes mellom kategoriene. Gruppen brukte Trello som et planleggingsverktøy og hjelpemiddel for å holde flyt i Scrum-prosessen. Ved å se på Trello-kort som en oppgave og lage de tre kategoriene ”Pågår”, ”Fungerende” og ”Ferdig” for å indikere tilstanden til oppgavenene kunne

	Native	React Native	Oracle APEX	Oracle JET	Xamarin	PhoneGap	Ionic
Visual Builder	Ja	Nei	Ja	Ja	Ja	Nei	Nei
God dokumentasjon fra utvikler	Ja	Ja	Ja	Ja	Ja	Nei (kun noen tutorials)	Ja
Stor brukerbase	Ja	Ja	Nei	Nei	Noe	Nei	Noe
Språk	Java og Swift	Javascript og React	Javascript CSS og HTML	Javascript CSS og HTML	C#	Javascript CSS og HTML	Javascript, CSS og HTML
Støtter mer enn en platform per kodebase	Nei	Ja	Ja	Ja	Delvis, noe spesifikk kode trengs	Ja	Ja
Kan bruke telefonspesifikke funksjoner	Ja	Eksisterer mange plugins	Nei, ikke hvis det ikke støttes av nettleseren	Bruker Cordova plugins	Ja, men kan trenge spesifikk kode for hver platform	Bruker Cordova plugins	Bruker Cordova plugins
Pek og klikk-editor som hovednavigasjon ved koding	Nei	Nei	Ja	Mulig	Nei	Nei	Nei
Egen Mac trengs for å kunne lage iOS-apper	Trenger	Muligens	Trenger ikke	Trenger ikke	Trenger	Trenger ikke	Trenger ikke
Egen Mac trengs for å kunne kompilere iOS-apper	Trenger	Trenger på original men ikke via Expo	Trenger	Trenger	Trenger	Trenger	Trenger

Tabell 1: Sammenligning mellom de forskjellige rammeverkene.

gruppen enkelt fordele oppgaver på medlemmene og ha god kontroll over hvilke oppgaver som er ferdige. For å bruke Trello sammen med Scrum valgte gruppen også å oprette en kategori for ”Product Backlog” og ”Sprint Backlog”. ”Product Backlog” ble hele tiden oppdatert med oppgaver som måtte utføres og ved starten av hver sprint ble utvalgte oppgaver flyttet over til ”Sprint Backlog” slik at gruppen til en hver tid visste hvilke oppgaver som måtte prioriteres.

3.1.5 Oracle database og SQL-developer

Som database bruker gruppen en Oracle-database da dette er databasesystemet EVRY bruker hos seg. For å kommunisere med databasen så brukes et program fra Oracle som heter SQL-developer. Dette programmet lar deg både opprette, endre og slette ting i databasen. I tillegg kan man gjøre mye annet som å implementere ”Oracle REST Data Services” (ORDS), som du kan lese om under.

En Oracle-database er en relasjonsdatabase, altså en database som lagrer data som tabeller. Den største forskjellen på Oracle sin database og andre relasjonsdatabaser slik som MySQL er at Oracle har et utvidet SQL-språk kalt PL/SQL. Dette språket lar man skrive SQL med utvidete muligheter som if-setninger og variabler.

3.1.6 Oracle REST Data Services

”Oracle REST Data Services”, eller ”ORDS” [8], er en tjeneste som lar deg enkelt lage REST-endpoints. ORDS er altså det samme som for eksempel Fusio [9]. Forskjellen er imidlertid at ORDS kun er mot Oracle-databaser, mens Fusio støtter flere databaser og andre måter å få info ut på.

Grunnen til at vi valgte ORDS var fordi det både var implementert på serveren allerede og fordi vi kan administrere API-endpointene direkte i SQL-developer. Les mer om implementasjonen av apier opp mot databasen i del 5.2.

3.1.7 Fullcalendar

Oracle Jet har ingen kalender-modul. Dette betyr at gruppen trengte å finne kalender på en annen måte. Løsningen ble Fullcalendar. Dette er et jQuery-basert bibliotek med mange muligheter til å modifisere slik at det blir en fullt integrert del av appen.

Grunnen til at gruppen valgte å bruke Fullcalendar var fordi man for det første hadde mange valgmuligheter med både utseende og andre endringer. I tillegg så var det oppskrifter på nett om hvordan man kunne lage Oracle Jet modul med Fullcalendar, som gjorde at vi visste at det ville fungere, selv om vi i første omgang ikke laget en egen fullverdig modul. Det var også bra at utseendet var noe som passet med resten av Oracle Jet.

En ting man kan gjøre er å legge til egne knapper i toppen og bunnen av kalenderen og samtidig bestemme hvor de ferdige knappene skal være samtidig som man angir de modifiserte knappene. Dette gjør at man på en helt naturlig måte kan legge til en egen filtreringsknapp der man kan be kalenderen oppdatere seg når man har lagt til en ny filtrering. Les mer om oppsettet av kalenderen i del 5.3.11.

3.1.8 Moment.js

Moment.js er et datobibliotek som kreves av Fullcalendar for å fungere. Det gjør det enklere å jobbe med datoer i javascript. Moment.js er nødvendig i Fullcalendar, men gruppen har valgt å bruke den andre steder også.

Moment.js lar deg enkelt for eksempel definere en dato og si i hvilket format og hvilken tidssone du vil vise tiden i. Dette gjøres for eksempel ved å skrive ”Europe/Oslo” for norsk tidssone.

Siden gruppen bruker Moment.js på mange av stedene gruppen tidligere brukte javascript sin innebygde dato har vi klart å gjøre koden mer lesbar. Koden er mer lesbar da den innebygde dato-klassen i javascript ikke klarer å formatere og forandre på datoen uten mye dårlig kode. Men siden Moment.js har klart å gjøre dette til en eller to kommandoer gjør dette koden mer lesbar.

3.1.9 Toastr

Toastr er et Javascript-bibliotek som lar man lage såkalte Toasts", som er meldinger som kommer på skjermen og går vekk igjen etter en viss tid, og som ikke forhindrer annen aktivitet.

Toastr brukes i appen der det lagres, endres eller slettes noe. I tillegg bruker vi det på feilmeldinger.

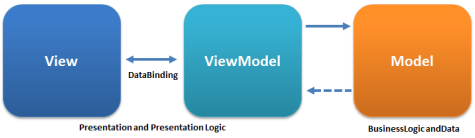
Biblioteket støtter både overskrift, melding og spesifikke innstilinger per melding, i tillegg til globale innstillinger. Utseendet på meldingene passer godt til Oracle Jet. De forskjellige meldingstypene er suksess (grønn farge), info (blå), advarsel (gul) og errormelding (rød).

Blant instillingene man kan endre på er tiden meldingen skal vises, om man skal vise en nedtelling (som en bar) og hvor på skjermen den skal vises, i tillegg til mange flere innstillinger.

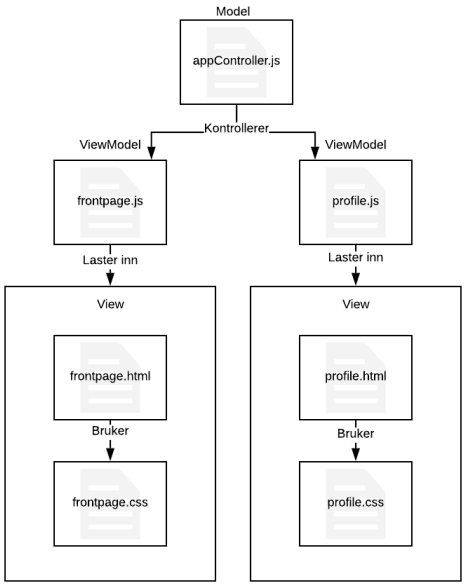
3.2 Arkitekturmønster

3.2.1 Model-View-Viewmodel

Oracle Jet støtter et arkitekturmønster som kalles ”Model-View-ViewModel” [10], heretter referert til som ”MVVM”. Dette mønsteret blir brukt for å separere koden for logikk og funksjonalitet fra koden som bestemmer applikasjonens utseende. De tre delene av MVVM referer til ulike deler av applikasjonen. ”Model” omhandler applikasjonens logikk, som for eksempel hvordan ulike sider blir vist frem for brukeren. Det er også i Model at alle data som skal lagres lokalt på en enhet bli definert. I denne bacheloroppgaven er koden til Model skrevet i javascript. ”View” refererer til koden som bestemmer utseendet på alt som blir vist til en bruker av applikasjonen. For dette har gruppen vår brukt HTML og CSS. Den siste delen, ”ViewModel”, er, som vist i figur 7, et bindeledd mellom Model og View. Hvert View har sin egen ViewModel som styrer logikken for det spesifikke View’et. All kode for ViewModel er også skrevet i javascript.



Figur 7: Hvordan MVVM henger sammen[11]



Figur 8: Eksempel på filstruktur som bruker MVVM

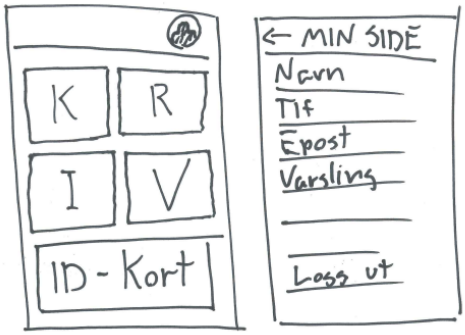
I figur 8 vises koblingen mellom filene til forside-siden og profil-siden og hvilke roller de har i MVVM strukturen. For oversiktligheten sin skyld er HTML- og CSS-kode separert i ulike filer, men hvert par av disse skaper tilsammen et View. Som nevnt ovenfor har et View en tilhørende ViewModel som styrer View'et. For at en MVVM struktur skal være effektiv er koblingen mellom View og ViewModel er veldig tett. Et av virkemidlene som gjør koblingen tett er javascript biblioteket "Knockout.js" som åpner mulighetene for å lage en direkte kobling mellom variabler. Direkte koblinger gjør at View og ViewModel kan oppdatere og bruke hverandres variabler, dette er nyttig hver gang applikasjonen skal håndtere inndata fra en bruker. Under vises et eksempel på hvordan hendelsesfoløpet, når en bruker prøver å logge inn, endrer seg med bruk av direkte koblinger.

Uten direkte koblinger

1. Bruker fyller ut brukernavn og passord
2. Verdier blir lagret i View
3. Bruker trykker på "Logg Inn"-knappen
4. ViewModel henter ut verdier fra View
5. ViewModel sjekker verdier opp mot databasen

Med direkte koblinger

1. Bruker fyller ut brukernavn og passord
2. Verdier blir lagret i View og ViewModel på grunn av direkte koblinger
3. Bruker trykker på "Logg Inn"-knappen
4. ViewModel sjekker verdier opp mot databasen



Figur 9: Tidlig tegning av forsiden.

3.3 Brukergrensesnitt

3.3.1 Plan for brukergrensesnitt

Som nevnt i 2.4.1, så inneholdt ikke oppgaven noe forslag til brukergrensesnitt, men gruppen hadde tilgang til både FRIDAs nettside og den gamle appen. Det ble enighet om å designe et nytt brukergrensesnitt, med noe inspirasjon fra kildematerialet. Målet var at både erfarne og nye brukere av FRIDA enkelt skulle kunne navigere appen uten problemer. Det var dermed viktig for oss å beholde samme knapper på forsiden som den tidligere appen. Selv om utseendet endret seg fullstendig, ble den samme teksten fremdeles vist på alle knappene, i tillegg til nye ikoner. Det var også viktig at brukeren ikke måtte trykke på mange forskjellige knapper for å finne øsnket funksjon, og at det til en hver tid var lett å komme tilbake forsiden.

Designprosessen startet med å tegne opp alle sidene i applikasjonen for hånd. Alle gruppens medlemmer jobbet sammen om dette, og ble enige om hvordan hver side skulle se ut. Figur 9 viser den originale tegningen av forsiden. Etter alle sidene var ferdige ble arkene scannet og gruppen startet å lage en prototype av appen. Tjenesten Justinmind ble brukt til å lage en prototype som viste omtrent hvordan appen skulle se ut og inneholdt knapper som navigerte sidene. [12] Etter prototypen var ferdig holdt gruppen et møte med EVRY hvor både tegninger og prototype ble presentert. Under møtet ble begge parter enige om hva som funket og hva som kunne forbedres. Deretter startet utviklingen.

Den samme prosessen ble brukt da appen ble utvidet til å inkludere administratorer, med unntak av prototypen og møte med EVRY. Disse ble utelatt da gruppen hadde begrenset tid igjen til prosjektet og grensesnittet ville i stor grad likne på det som allerede hadde blitt utviklet.

3.3.2 Faktisk brukergrensesnitt

Brukergrensesnittet endte opp med å likne veldig på de originale tegningene. FRIDAs grønnfarge ble tatt i bruk som et gjengående tema i hele appen, mens sort eller hvit tekst ble tatt i bruk avhengig av hvor det ble brukt og hvor stor teksten var. I siste utgave av appen, må brukeren maksimalt trykke på tilbakeknappen tre ganger for å komme tilbake til forsiden, noe gruppen er fornøyd med. Mer detaljert beskrivelse av hver side finnes under Funksjonalitet i kapittel 5.

3.3.3 Problemer med brukergrensenitt

Under utvikling oppstod det diverse problemer med å designe appen etter planen. For å blant annet lage knapper ble Oracle JETs egne knapp-elementer ”oj-button” brukt, da disse gjør det lettere å kommunisere med Javascript koden. Disse elementene kom med egen innebygd CSS kode, som til tider kunne være vanskelig å overstyre. I første omgang var det vanskelig å gi knappene den størrelsen og posisjonen de skulle ha, men dette løste seg så fort gruppen lærte seg Oracle JETs egen syntax. Fargen til knappene var derimot ikke like enkel å endre, noe EVRY heller ikke klarte å hjelpe oss med i første omgang. Også andre, uønskede, egenskaper med knappene var krevende å endre, blant annet at knappene endret farge når en holdt pekeren over dem og når de ble trykket på. Gruppen fant omveier en kunne bruke for å endre alle elementene til egne ønsker, men fant aldri ut om JET tilbyr en egen måte å gjøre dette på.

3.4 Universell utforming

Vi prøver å følge universell utforming så mye som mulig. Det er også et krav fra norske myndigheter at alle nettsider og apper som skal være tilgjengelig for publikum har et minimumskrav om Nivå A.

På nettsidene til Difi kan vi se forskjellige kriterium som må støttes for dette. I denne seksjonen vil vi prøve å vise til hvordan vi har fulgt universell utforming i appen etter kriteriene listet opp i [13] for kriteriene som er relevante for appen.

Men generelt sett så kan vi presisere at Oracle Jet sine elementer er universelt utformet på nivå AA [14], så hvis vi bruker Oracle Jet elementene korrekt så skal dette være fikset.

3.4.1 Ikke tekstlig innhold (Nivå A)

Det er ikke mye av appen som ikke har tekst, men når det gjelder tekst i inputelementer så er teksten som sier hva som skal være i elementet lagt til som en ”oj-label”-komponent. Denne komponenten støtter koblingen i HTML5 som gjør at det er lettere for skjermlesere og andre hjelpemidler å lese teksten som en del av feltet. I tillegg så har tekstfeltene placeholdere med informasjon. Dette både fordi designet for iOS gjør at det nesten er umulig å se tekstelementene uten placeholdere, og fordi det vil gjøre det lettere for skjermlesere å bruke.

3.4.2 Informasjon og relasjoner (Nivå A)

Når det gjelder overskrifter så er overskriftene i vår hybridapplikasjon lagt til i headeren. Denne headeren er automatisk generert av Oracle Jet, slik at så lange overskrift-delen av headeren ikke endres så skal den følge standarden.

Når det gjelder annen tekstinndeling så har vi sjelden mye tekst, slik at mellomoverskrifter er skjeldent nødvendig.

Når det gjelder skjema så er det som beskrevet over lagt til koblinger mellom label og inputelementer i tillegg til placeholdere, slik at dette blir enklere for skjermlesere og andre hjelpemidler å lese korrekt.

3.4.3 Meningsfylt rekkefølge (Nivå A)

Rekkefølgen på hvordan ting presenteres innholdsmessig er så enkel at det bør være enkelt for skjermlesere å lese innholdet korrekt. I tillegg så er elementer som er listet opp sortert på en meningsfull måte.

3.4.4 Sensoriske egenskaper (Nivå A)

I appen er tekst koblet til de aller fleste elementer.

3.4.5 Bruk av farge (Nivå A)

Farger på knapper og andre elementer er kun en visuell nytelse. Meningen med knappene og andre elementer er forklart med tekst i tillegg.

3.4.6 Kontrast (minimum, Nivå AA)

Det er meningen at appen skal ha god kontrast mellom tekst og bakgrunn.

3.4.7 Endring av tekststørrelse (Nivå AA)

Hvis tekststørrelse endres til det dobbelte, altså 200%, så kan det hende at noe tekst wrappes siden dette er på en mobil platform, men siden appen har mulighet for scrolling på alle sider, så vil ingen funksjonalitet gå tapt.

3.4.8 Bilder av tekst (Nivå AA)

Appen bruker kun bilde av tekst i logoen, og logoen anses som nødvendig, og er derfor ikke omfattet av kravet.

3.4.9 Sidetitler (Nivå A)

Gruppen mener at sidetitlene i appen er tydelige.

3.4.10 Fokusrekkefølge (Nivå A)

Gruppen mener at sider kommer i den nødvendige rekkegfølgen som passer. For eksempel så må man først gå inn på ”Mine interesser” for å kunne gå videre inn for å legge til nye interesser.

3.4.11 Formål med lenke (i kontekst, Nivå A)

Alle knapper er tydelig merkert enten med tekst, bilde eller begge deler.

3.4.12 Flere måter (Nivå AA)

Appen har ikke alltid flere måter å gjøre en ting på, men appen er ikke stor, og vi ser det derfor ikke like hensiktsmessig å ha flere måter å navigere seg rundt på som store nettsider ville hatt det.

3.4.13 Språk på siden (Nivå A)

Språket i lang-ttributtet i html-siden er definert til å være norsk.

3.4.14 Fokus (Nivå A) og inndata (Nivå A)

Endring av fokus eller endring i et skjemafelt vil ikke automatisk endre innstillingen. Dette gjelder ikke for søkefelt, men her ser gruppen for seg at dette ikke omfattes av disse kravene i og med at automatisk søk fins på de fleste sider.

3.4.15 Konsekvent identifikasjon (Nivå AA)

Gruppen har som mål at sidene skal være så like som mulig. For å få til dette går gruppen igjennom sidene og sørger for at de er like etter at de er laget.

3.4.16 Identifikasjon av feil (Nivå A) og forslag ved feil (Nivå AA)

Appen informerer brukerne at noe feilet, for eksempel at innlogging feilet fordi brukernavn eller passord var feil. Vi informerer også for eksempel om at man kan prøve igjen hvis det ikke gikk an å lagre en endring for eksempel.

3.4.17 Parsing (oppdeling, Nivå A)

Det er nesten umulig å ha store kodefeil i koden ettersom at det i Oracle JET, spesielt i hybrid-applikasjoner, men også webapplikasjoner, ikke vil vises hvis det er en ørliten småfeil. Dette har ført til mange irritasjonsmomenter under utvikling ettersom flere slike errorer er så små at de ikke en gang blir oppdaget av de fleste debuggere som burde oppdaget dem.

4 Utviklingsprosess

4.1 En vanlig arbeidsuke

En vanlig arbeidsuke så slik ut:

- Mandag: Andre timer. Hovedsaklig ble denne dagen brukt til å jobbe med de andre fagene gruppen hadde. I tillegg tok gruppen noen korte møter noen uker. Dette var satt til etter at vi var ferdige med timene den dagen, eller da vi startet første timen om vi ikke hadde noen timer denne uken. Da ble det som regel planlagt noe i forhold til møtet med EVRY dagen etter.
- Tirsdag: Møte og jobbing hos EVRY. Møter på Gjøvik Skysstasjon for å ta bussen til Brummundal 08.25. Da var gruppen hos EVRY i Brummundal ca. 09.30. Avsluttet som regel dagen ved å ta bussen tilbake til Gjøvik litt over 15.30.
- Onsdag: Møte med veileder og jobbing på skolen. Møtes 10.00 i Atriet for å planlegge hva vi skal snakke med veileder om denne dagen. Gikk deretter i møte med veileder 10.30. Etter møtet plasserte vi oss på et prebooket rom og jobbet sammen. Avsluttet 16.00.
- Torsdag og Fredag: Jobbing på skolen. Jobbet sammen på et prebooket rom fra 09.00 til 16.00.

4.2 Sprintene

Sprintene var, som tidligere nevnt, to arbiedsuker lange, og ble innholdsvis definert i starten av prosjektet. Den første sprinten fokuserte i tillegg på å bli konfortable med Oracle JET. Figur 10 viser et utsnitt av Trelloen til gruppen. I starten av hver sprint ble alle avtalte punkter flyttet fra 'Backlog' til 'Sprint Backlog', og gruppens medlemmer kunne deretter velge oppgaver fritt. Frem til sprint fire fulgte gruppen planen ganske nøyaktig, men da første testrunde måtte utsettes en uke ble også deler innholdet i sprint fire utsatt. Dette førte til en ukes forsinkelse ut utviklingsfasen.

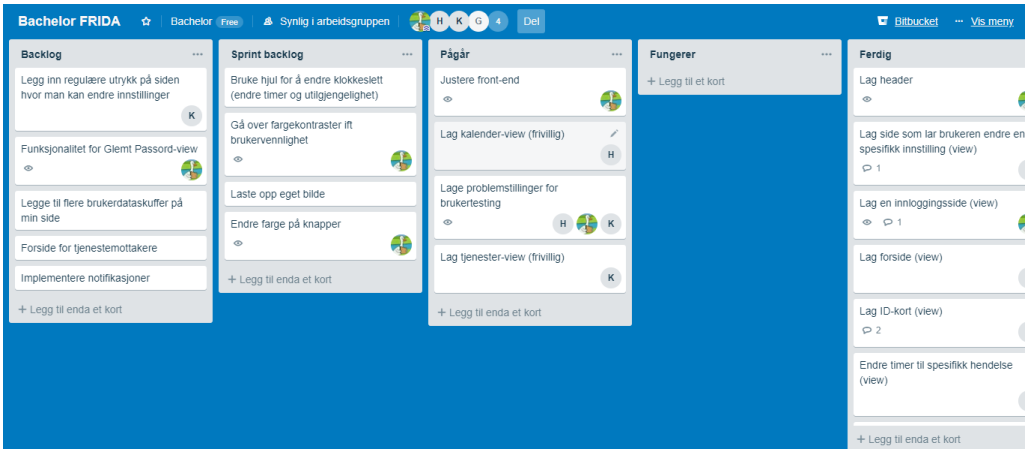
4.3 Sikkerhet

Under utviklingsprosessen har sikkerhet vært noe gruppen har vært klar over må gjøres. I del 2.3 ble det nevnt noen krav vi hadde til sikkerhet. Under vil vi ramse opp hva statusen til disse er, hva annet som har blitt gjort og hva som har blitt satt på vent (og som det må gjøres noe med), i forhold til å gjøre appen så sikker som mulig. I tillegg vil gruppen fortelle hva som er sensitiv informasjon. Vi viser også et misuse-case diagram for å vise noen metoder å hacke systemet på og noen løsninger på dette.

4.3.1 Sensitiv informasjon

Databasen inneholder en del sensitiv informasjon om de forskjellige brukerne av systemet som kontaktinformasjon, adresser og navn. Dette gjør at hvis sikkerheten ikke er på topp rundt databasen og der informasjon går ut igjennom APIet så blir dette en sikkerhetsrisiko.

Testdatabasen som ble gitt av EVRY var en ren kopi av deres database med ekte brukere.



Figur 10: Utsnitt av Trello under sprint tre.

Det var også grunnen til at gruppen ba om å undertegne på en egen taushetsplikt, slik at alle lover og regler ble fulgt korrekt.

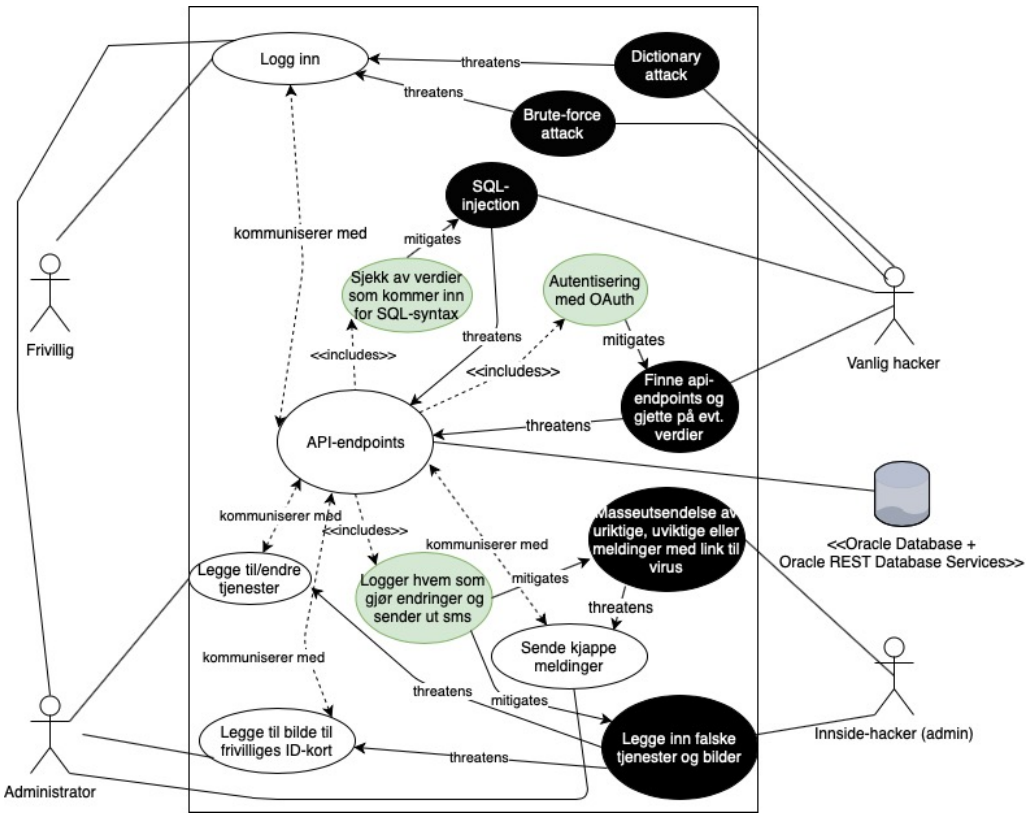
4.3.2 Misuse-case diagram

Dette systemet er hovedsaklig basert på api-endpoints for å få informasjon ned fra systemets database. Derfor er de fleste sikkerhetsrisikoer rettet mot dette apiet. Derfor er api-endpoints tegnet inn som hovedkilden hackere kan prøve seg på i figur 11. I tillegg er det lagt inn at en vanlig hacker kan prøve å hacke seg inn ved å prøve forskjellige kombinasjoner med brukernavn og passord.

I tillegg til å kunne være en vanlig hacker så kan også en innside-hacker med admin-rettigheter kunne gjøre uventede ting. Dette er derfor også tatt med.

Under listes de forskjellige misuse-casene opp og beskrives nærmere med hva som kan løse disse problemene.

- Vanlig hacker: Kan bruke Brute Force Attack for å prøve å logge inn. Brute Force Attack betyr at hackeren prøver å gjette passordet ved å bruke et automatisert program som genererer forskjellige passord [15]. Dette kunne forhindres ved at appen sjekker antall ganger man prøver å logge inn, men dette er foreløpig ikke lagt inn da det er mulig mye av denne delen av appen må forandres når passordsystemet til EVRY blir implementert. I tillegg kan man kreve sterke passord da dette vil gjøre det vanskeligere å autogenerere passord.
- Vanlig hacker: Dictionary Attack er en form for Brute Force Attack der man prøver kjente passord slik som ”123”, ”qwerty”, ”ManU”, etc [15]. For å forhindre dette angrepet fra å lykkes er det viktig at passordene er unike. Det anbefales derfor å bruke en passorddatabase for å kunne lagre unike passord om brukeren har vanskelig for å lage unike passord.
- Vanlig hacker: Kan bruke SQL-injection mot databasen. SQL-injection kan prøves i alle felt som sender tekst til databasen via SQL. Dette er, som sagt i 4.3.3, løst ved automatisk beskyttelse via ORDS.
- Vanlig hacker: Kan finne api-endpoints og gjette på korrekte verdier i felter som for



Figur 11: Et Misuse-case diagram med noen løsninger på å sikre systemet.

```
1 var EmpModel = oj.Model.extend({urlRoot: self.serviceURL, parse:
      self.parseTweet, idAttribute: 'id'});
2 var Emp = new EmpModel();
```

Listing 4.1: Kodesnutt for OAuth modell og samling [1]

eksempel id til brukere. Dette problemet er løst ved å ha autentisering på REST-apiet med OAuth. Les mer om dette i del 4.3.3.

- Inside-hacker (admin): Masseutsendelse av uriktige, uviktige eller meldinger med link til virus. Dette er et problem som gjerne kan oppstå ved at noen misbruker sin stilling. Dette kan enkelt løses ved at systemet logger hvem som sender ut SMS slik at denne personen kan sperres fra systemet.
- Inside-hacker (admin): Legge inn falske tjenester og bilder. Dette er et problem som gjerne kan oppstå ved at noen misbruker sin stilling. Dette kan enkelt løses ved at systemet automatisk logger hvem som lager og endrer ting i systemet. Da er det mulig å finne ut hvem denne personen er og sperre denne personen ute av systemet.

4.3.3 API

Som nevnt over så har vi hele tiden vært klar på at APIet må ha et sterkt sikkerhetsnett rundt seg før appen blir utgitt. APIet blir kjørt via ”Oracle REST database service” (ORDS), som det kan leses mer om i 3.1.6. Siden dette er en tjeneste hos et stort firma kan vi regne med at det har blitt gjort noe med SQL injection via JSON. Å ha en fiks for dette var et av sikkerhetskravene vi hadde.

I ORDS har vi også mulighet til å beskytte APIet med autentisering. Dette har vi derimot ikke gjort noe med foreløpig. Grunnen til dette er at gruppen, etter å ha lest hvordan man kan legge til autentisering, fant ut at dette var så vanskelig å finne korrekt måte å gjøre det på, at det var bedre å vente og la EVRY legge på dette senere. Grunnen til at gruppen valgte å vente med dette var blant annet fordi det var usikkert om hva som var den beste måten å gjøre dette på, og muligheten av at EVRY ville endre på måten når de endrer innlogging fra det vi evt. ville lagt på. Derfor fant vi ut at det var bedre å vente enn å bruke tid på å legge til noe som kunne eller ikke kunne brukes.

Da gruppen snakket med EVRY om dette sa de at de skulle se på dette på serversiden for oss. De fikk problemer med implementeringen og har satt dette på vent til de er sikre på at de ikke skaper problemer for oss. Derfor har vi avtalt med dem om at vi ikke trenger å tenke på å implementere dette. Vi har derimot siden sørget for at det enkelt skal være mulig å legge til headere til alle ajax-kall da det er det som brukes i OAuth til å autentisere.

Hvordan implementere OAuth i ORDS på klientsiden (i Oracle JET)

Hvis OAuth er lagt til på serversiden i ORDS, som man kan lese om under, så kan man ifølge [1], implementere OAuth i applikasjonen på denne måten:

1. Legg til ”ojs/ojmodel” som en avhengighet på siden, da den inneholder OAuth-klassen.
2. Lag et OAuth-objekt på denne måten: ”self.OAuth = new oj.OAuth();”.
3. Deretter lager man en modell og samling som vist i kodesnutt 4.1.
4. Deretter kan man bruke linjene i kodesnutt 4.2, som ferdiggjør en spørring.

Implementeringen via denne klassen fant gruppen veldig sent i prosessen, og som nevnt tidligere har vi antatt å implementere OAuth som en egen header manuelt. Å implementere

```
1 var credentials = {client_id: <some_id>, client_secret:
      <some_secret>, bearer_url: <some_url>}
2 self.OAuth.setAccessTokenRequest(credentials);
3 var EmpCol = oj.Collection.extend({url: self.serviceURL, model: Emp,
      oauth: self.OAuth});
```

Listing 4.2: Kodesnutt for slutten på OAuth [1]

headeren manuelt har gruppen tenkt at vi skulle gjøre da andre guider om dette temaet har valgt å gjøre det slik. Siden koden gruppen har skrevet er basert på denne fremgangsmåten og ikke guiden vi har vist over, så er muligens den beste koden å legge headeren inn manuelt i våre ferdiglagde ajax-kall. Denne headeren er i tilfelle ”Authorization: Bearer «Token»” [16], der ”«Token»” er en midlertidig token man får når man kaller et spesifikt api-kall for å logge inn i api’et.

Hvordan implementere OAuth i ORDS på serversiden

Som nevnt over ble vi enige med EVRY om at de skulle se på ORDS på serversiden. Dette ble delvis bestemt fordi det var vanskelig å opprette alt som måtte til for dette på serversiden. Ivertfall i databasen EVRY ga oss. Siden, som nevnt tidligere, EVRY også fikk problemer med å implementere dette betyr det at dette ikke var så lett som det kan se ut. Under kommer vi med et kort sammendrag av hva man må gjøre i følge guiden til [2].

1. Opprett en database med noen api-endpoints.
2. Opprett en eller flere roller og privilegier. Dette kan enten gjøres med noen PL/SQL-statements eller via visuelle former i SQL-developer. Ett privilegium er assosiert med en rolle.
3. For å beskytte et api-kall, så mappes et privilegium med api-kallet. Dette kan også gjøres enten direkte i PL/SQL eller via SQL-developer sine visuelle former.
4. Når dette er gjort går det ikke an å koble seg til apiet lenger uten autentisering. Enda har man ikke definert hvilken autentiseringsmetode man vil bruke enda, men kan velge mellom ”Basic”-autentisering og OAuth 2. Instruksjonene videre baserer seg på at man har valgt OAuth 2.
5. I OAuth 2 er det flere autentiseringsmetoder å velge mellom. Blant annet ”Client Credentials” og ”Authorization Code”. Blant alle man kan velge mellom er det ”Client Credentials” som passer best i denne appen da denne ikke krever noen spesiell bruker-interaksjon, som betyr at man, som man gjør det i ”Authorization Code”, må besøke en nettside. Derfor baserer resten av instruksjonene på at man har valgt å bruke ”Client Credentials”.
6. ”Client Credentials” er en prosess i to steg der man lager klienter (”clients”) som man kan logge seg på. Hvordan man lager en klient i PL/SQL kan man se i kodesnutt 4.3. Da autogenereres det en ”client_id” og en ”client_secret”. Man kan enten ha en klient for admin og en for frivillig, eller lage en klient for hver bruker om man vil det.
7. Deretter må vi assosiere klienten med rollen som har de korrekte privilegiene, slik at frivillige ikke kan gjøre ting som bare admin kan og omvendt. En PL/SQL-kode for dette kan sees i kodesnutt 4.4.
8. Deretter må man bruke klientens id og ”secret” for å kalle et api-endpoint på urlen ”oauth/token”. Da vil man ved suksess få en token tilbake.
9. Da kan man putte den token man fikk ved forrige kall i alle eksisterende kall. Over så

```
CONN testuser1/testuser1@pdb1

BEGIN
  OAUTH.create_client(
    p_name            => 'Emp Client',
    p_grant_type      => 'client_credentials',
    p_owner           => 'My Company Limited',
    p_description     => 'A client for Emp management',
    p_support_email   => 'tim@example.com',
    p_privilege_names => 'emp_priv'
  );

  COMMIT;
END;
```

Listing 4.3: PL/SQL kode for å lage en OAuth-klient [2]

```
BEGIN
  OAUTH.grant_client_role(
    p_client_name => 'Emp Client',
    p_role_name   => 'emp_role'
  );

  COMMIT;
END;
```

Listing 4.4: PL/SQL kode for å assosiere en OAuth-klient med en OAuth-rolle [2]

ble det forklart en måte å gjøre dette på klienten, men som sagt over vil en like enkel, om ikke enklere metode være å legge til headeren direkte i hvert ajax-kall. Det eneste man da må være oppmerksom på er at token etterhvert vil gå ut, og at man da må skaffe en ny en ved å for eksempel automatisk logge ut.

4.3.4 Brukerinput

Når det gjelder input fra brukere, så burde gruppen muligens brukt noe lignende funksjonen htmlspecialchars(..) gjør i PHP. Grunnen til at dette ikke har blitt gjort er hovedsaklig fordi det ikke eksisterer en innebygd funksjon i javascript. Gruppen har senere funnet ut at en funksjon som den vist i kodesnutt 4.5, kan funke.

Dette har vi derimot ikke implementert da vi ser at vi ikke hadde tid til å gå over dette. Gruppen ser heller ikke dette som det viktigste å gjøre, da andre sikkerhetsalternativer er viktigere. I tillegg vil alt som blir lagret i databasen gå igjennom ORDS, så da vil noe av det

```
function escapeHtml(text) {
      var map = {
        '&': '&amp;',
        '<': '&lt;',
        '>': '&gt;',
        '"': '&quot;',
        "'": '&#039;'
      };

  return text.replace(/[&<>"']/g, function(m) { return map[m]; });
}
```

Listing 4.5: Mulig Javascript versjon av htmlspecialchars(..) [3]

```
var rootViewModel = ko.dataFor(document.getElementById('globalBody'));
//If userId is undefined, go back to login-page:
if (rootViewModel.userId() == undefined) {
    oj.Router.rootInstance.go('login');
    return
}
```

Listing 4.6: If-setning for å sjekke om man har mistet info om innlogget bruker og "logg ut" om dette er tilfelle

alvorligste bli sjekket ut.

4.3.5 Automatisk utlogging når app glemmer informasjon om brukeren

På alle sider som man må være innlogget på så er det satt opp en sjekk på at man vet hvem som er innlogget, og går til innloggingssiden om dette ikke er tilfelle. Koden for dette kan man se i kodesnutt 4.6. Denne if-setningen sjekker om variabelen som lagrer bruker-id ikke har en verdi, eller altså er undefined, og hvis den ikke har en verdi lengre logger vi ut. På login-siden må man da logge inn igjen for å kunne få en verdi i variabelen igjen.

4.4 Tekniske problemer som kunne vært unngått med bedre kommunikasjon eller planlegging

4.4.1 Bytting av servere hos EVRY

Godt ut i prosessen, samme dag som gruppen hadde avtalt med EVRY og Hamar frivillighetssentral å teste med noen frivillige hos dem, så byttet EVRY servere. Dette gjorde at når en på gruppen tilfeldigvis sjekket appen samme dag som vi skulle teste, så funket ikke tilkoblingen til REST apiene.

Selv om EVRY hadde overført databasen til de nye serverne, så hadde de ikke overført apiene. I tillegg så hadde de endret urlen til apiene slik at selv om apiene var overført så hadde gruppen måtte bytte ut alle urlene. Siden gruppen på det tidspunktet ikke hadde laget en base-url variabel så tok dette en stund å fikse. Men da laget gruppen denne variabelen slik at man bare trenger å endre urlen ett sted når det evt. skal bli satt ut i produksjon.

I tillegg tok gruppen seg tid til å lage et mock-api for å ha mulighet til å vise appen hvis dette skulle skje igjen foran en demo.

Mens det å overføre apiene ble fikset på dagen siden EVRY startet opp den gamle serveren slik at det var mulig å overføre apiene fra den gamle serveren til den nye, så var det enda flere problemer som ventet. Problemet som ventet nå var at den versjonen av ORDS som EVRY hadde på den nye serveren hadde et problem som gjorde at vi ikke fikk tilgang til api'ene. Det tok flere dager før EVRY hadde oppdaget hvorfor vi ikke fikk tilgang til apiene og fikset dette, da det var et problem på deres servere. Problemet var ifølge Oracle "Insufficient Privileges, using RESTFul Services" [17]. EVRY fikset dette med en midlertidig løsning gitt av Oracle, mens de senere ville oppdatere til nyeste versjon av ORDS som ikke har denne feilen.

5 Implementasjon

5.1 Status på de operasjonelle kravene

I denne delen beskrives statusen til de operasjonelle kravene listet opp i del 2.2. Om de ikke har blitt gjennomført har det blitt gitt en begrunnelse på hvorfor.

5.1.1 Skal fungere med det eksisterende FRIDA-systemet

Dette har blitt løst ved at vi har brukt en eksakt kopi av databasen til det eksisterende systemet med unntak av at vi har en annen innlogging, som man kan lese mer om i 5.3.1. Når denne delen er byttet til annen innlogging er det enkelt å bytte til produksjonsserver. Les mer om testdatabasen og hva som må gjøres før appen kan bli satt ut i produksjon i kapittel 6.

5.1.2 Oppetid på 99%

Dette er et krav gruppen ikke har kontroll over da det kommer an på databasen og serveren til EVRY, men er et krav gruppen har prøvd å kontrollere ved å ha sterke api-koblinger. Med sterke api-koblinger mener vi her for eksempel at det ikke skal være mulig å være innlogget uten at informasjonen som midlertidig lagres i variabler har korrekt innhold. Hvis dette ikke hadde blitt gjort kunne man tro at serveren var nede istedenfor at appen har glemt hvem som er innlogget.

5.1.3 Innlastningstid på et par sekunder

Dette har vært et ønske som desverre ikke har blitt kunnet være gjennomførbart i alle sammenhenger. Dette fordi internettets hastighet har vært en stor faktor her. Hvis man har normalt kjapt internett går dette kjapt, slik vi ønsker oss. Men desverre så har vi opplevd at spesielt ”Eduroam”, NTNU sitt trådløse nettverk, har vært alt for tregt. Da kan det faktisk gå rundt et halvt minutt før noe skjedde.

På grunn av at det i noen sammenhenger kan være et så tregt internett så burde gruppen muligens ha laget en indikator på at noe lastes, men dette har gruppen ikke hatt tid til. Men med et normalt kjapt internett eller lasting via mobilnettet vil dette ikke være et stort problem. Med mobilnett menes her hovedsaklig 4G og etterhvert 5G da 3G blir faset ut.

5.1.4 Appen skal funke på både iOS og Android

Som det står i blant annet 3.1.1, så ble Oracle JET valgt, som støtter hybride apper både iOS og Android, i tillegg til Windows Phone. Prosjektet ble satt opp til å støtte iOS og Android da vi ikke kunne sjekke om alt funket korrekt på en Windows Phone.

5.1.5 Appen skal være enkel å vedlikeholde

Siden det er Oracle JET som brukes så er det enkelt å overføre fra den ene kodebasen som lages og kjøre det på flere platformer via Cordova. Det er heller ikke mye systemfunksjonalitet, noe som gjør at det skal være få plugins som trengs, som gjør appen enkel. Oracle JET gjør det også enkelt å opprette nye sider da alt er laget per side, altså at det er en HTML-fil og en Javascript-fil per side.

5.2 Integrasjonen av API opp mot databasen

For å lage apier for å få informasjon ut fra databasen ble det som nevnt i 3.1.6, brukt tjenesten ORDS. For å lage et endpoint ble det brukt enten SQL eller PL/SQL. På et endpoint ble det faktisk kun brukt et valg i SQL-developer om å vise bildet som var lagret.

Som det også var nevnt tidligere ble SQL-developer brukt for å lage apiet. Det ble gjort ved å gå inn i en meny der man valgte hva outputten skulle bli. Dette var altså enten et media, slik som et bilde, slik det er brukt i et endpoint, eller et enrads output, eller muligheten for å returnere flere rader. Det er også ikke alle valg som er tilgjengelige til enhver handler (GET, POST, PUT og DELETE). For eksempel støtter GET både vanlig SQL og PL/SQL, mens POST kun støtter PL/SQL.

Når man velger å skrive enten PL/SQL eller SQL får man opp en tekstboks der man kan angi henholdsvis PL/SQL eller SQL-kode som skal kjøres når apiet kalles. Selv om alt dette skjer i SQL-developer visuelt, så blir en form for PL/SQL-kode kjørt i bakgrunnen når nye api-endpoints opprettes.

Dette betyr at koden for å få noe ut av databasen er henholdsvis vanlig SQL eller PL/SQL. Dette gjør det ikke noe vanskeligere å lage api-kall enn om det var programmert i for eksempel PHP. Snarere tvert i mot da den koden som i tilfellet ville vært i PHP er skrevet av noen andre på forhånd.

APIet har derfor enkelt kunnet hente informasjon fra databasen. APIet kan i sin tur enkelt bli kalt fra appen. For å se noen metoder vi har gjort dette på og tanker rundt valg av funksjon til dette kan leses om i 5.4.1.

5.2.1 Eksempel på SQL-kode for å definere et endpoint i SQL-developer

Et eksempel på en kodesnutt med vanlig SQL-kode for et GET-api-kall kan være kodesnutt 5.1. URL'en for denne kan for eksempel være "/tjenester/:startDato/:sluttDato". De to variablene :startDato og :sluttDato definerer her et tall som kan være forskjellig fra gang til gang når url'en kalles. Disse variablene blir referert til i begge TO_DATE(..)-funksjonene. TO_DATE(..)-funksjonen konverterer en tekststreng med et spesielt format om til DATETIME-format, slik at den kan brukes i sammenligning med variabelen. Dette api-kallet vil returnere alle tjenester fra og med datoen definert i variabelen ":startDato" og til og med datoen definert i variabelen ":sluttDato".

Den fulle koden som blir kjørt for å opprette et api-endpoint i bakgrunnen er ikke relevant å gå inn på da dette ikke er skrevet manuelt.

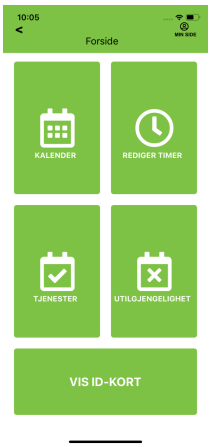
```
1 SELECT navn, dato, beskrivelse
2 FROM tjenester
3 WHERE dato >= TO_DATE(:startDato, 'DD_MM_YYYY') AND dato <=
      TO_DATE(:sluttDato, 'DD_MM_YYYY')
```

Listing 5.1: Kode for et fiktivt GET-kall

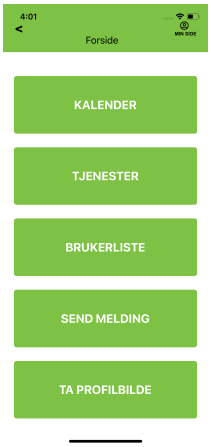
5.3 Funksjonalitet

5.3.1 Login

Login-siden er en helt vanlig login-side. Denne siden åpnes alltid når appen restartes, eller hvis appen på en eler annen måte har mistet hvilken id brukeren har. Siden har også en link til Glemt passord. Forsiden sender brukernavn og passord til databasen, og deretter fås en respons med brukerid hvis brukeren eksisterer. Hvis brukeren ikke eksisterer får du ikke



Figur 12: Den nye appens frivilligforside



Figur 13: Den nye appens administrator-forside

logget inn.

Det går an å sende brukernavn og passord sikkert fordi det brukes https, men hvis denne måten å sjekke om brukernavn og passord var korrekt skulle brukes i den ferdige appen, måtte appen muligens ha hashed passordet før det ble sendt til backend. Men siden login-metoden uansett vil bli skiftet ut med et helt annet system før det evt. blir tatt i bruk trengte ikke gruppen å gjøre login veldig sikkert.

5.3.2 Glemt passord

I glemt passord kan du spørre om nytt passord om du har glemt det. Hvis du ber om nytt passord blir et nytt passord autogenerert, med variabel lengde og med store og små bokstaver + tall. Brukeren skal så få dette nye passordet tilsendt, slik at man kan logge inn og evt. endre passordet til noe man lettere kan huske.

5.3.3 Forside

Forsiden ser litt forskjellig ut utifra om man er frivillig eller administrator

Frivilliges forside

Som vi kan se i figur 2 i starten av rapporten så er forsiden/menyen på dagens app en liste med små, like knapper. Dette gjør at den eneste måten å vite hva de forskjellige knappene på dagens app gjør så må man lese navnet på knappene eller huske hvilken linje knappen er på. i tillegg er knappene så små at en person med litt dårlig koordinasjon i fingrene, noe eldre frivillige for eksempel kan ha, kan fort bomme på riktig knapp.

For å designe og implementere frivillig-forsiden på best mulig måte, så bestemte gruppen seg, som vi kan se i figur 12, derfor å lage store knapper for alle knappene på forsiden slik at det skal være mye mindre sjanse for å bomme på knappen man vil trykke på. I tillegg så har de fleste knappene, i tillegg til tekst, også et bilde som gjør at det kan være lettere å huske hva knappen er til når man har lært seg hva bildene betyr. I tillegg så er knappene plassert slik at det kan være lettere å huske hvor hver ting ligger enn om det hadde vært en liste slik dagens app gjør det.

Dette betyr altså at vi har tatt inn designmomenter som gjør at appen følger universell

utforming. Les mer om dette i del 3.4.

Administratorforside

På administratorens forside, slik som man kan se på figur 13, så har gruppen valgt å ha en liste slik det er på dagens app. Dette gjør at det er lettere å fjerne og legge til funksjonalitet fordi man ikke trenger å tenke på å endre design, men det betyr at det forutsetter at administrator ikke trenger et superenkelt design for å manøvrere rundt.

Koble sammen forsiden og de andre delene av appen

I Oracle JET så brukes kommandoen ”oj.Router.rootInstance.go(page);”, der page er en string-variabel med gitt navn på siden som skal vises Dette er en enkel kommando, man må bare passe på å registrere alt korrekt på de korrekte stedene når siden opprettes.

5.3.4 Header

Den øverste baren som vises på nesten alle sidene i appen kalles headeren. Det er her man kan gå til ”Min-side” og gå bakover i appen på samme måte som er vanlig i IOS. Oracle JET har desverre ingen egen tilbake-funksjon når det gjelder å gå til en annen side. Den eneste funksjonen som eksisterer er ”oj.Router.rootInstance.go(page);”, som nevnt i del 5.3.3.

Dette betyr at tilbake-funksjonaliteten er noe gruppen måtte lage selv. Koden for dette kan sees i kodesnutt 5.2. Her kan man se at man ikke kan gå tilbake til login-siden. Det er også lagt inn en begrensning slik at man fra profilsiden går tilbake til forsiden. Dette er en ting som er lagt på for at man skal kunne gå direkte fra profilen til forsiden da dette kan være ønskelig mange ganger. Deretter sjekkes det også mot ganger der man skal hoppe over en side. Hvis man heller ikke skal gjøre dette kommer det til å skje en vanlig tilbakegang. For å kunne gå tilbake, som vi kan se på for eksempel linje 18 og 19, som er den vanlige tilbakegangen, at man først går til siden som var lagret på plassen tidligere i arrayet. Deretter blir de siste sidene pop’et av arrayet som lagrer hvilke sider som har blitt vist.

5.3.5 ID-kort

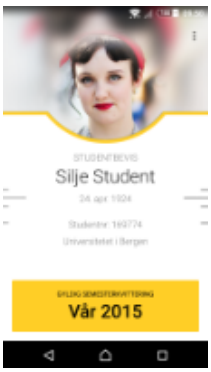
For å implementere visning av ID-kort, så fant vi ut at dette burde være laget slik at det er enkelt å se at ID-kortet er ekte. Dette betyr at vi måtte lage en eller annen funksjonalitet som viser at det ikke bare er et bilde hvor man har endret teksten og bildet slik at andre kan lage falske ID-kort. Som vi kan se av figur 4 i starten av denne rapporten, så har ikke dagens app en slik funksjonalitet. Dette betyr at dagens app har en sikkerhetsrisiko i forhold til at uønskede personer kan komme seg inn hos andre på gale premisser.

For å sørge for at dette ikke kan skje, så ble vi inspirert av studentbevisappen. Denne appen er bevis for at vi studenter er studenter. Som vi kan se av figur 14 så inneholder studentbevisappen et bilde av ansiktet til studenten, fødselsdato, studentnummer og studiested. I tillegg er det info om gyldig/ugyldig semesterkvittering.

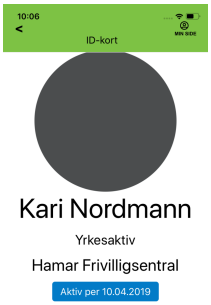
Trykker man på feltet for gyldig semesterkvittering så skjer en animasjon rundt bildet av studentens ansikt. Mer nøyaktig så går ansiktet nedover. Vi har blitt inspirert av dette og laget en animasjon som gjør at profilbildet i den nye FRIDA-appen, som er den svarte rundingen på figur 15, spinner rundt når man trykker på gyldig-feltet. I tillegg så er måten vi har satt opp ID-kortet til den nye appen, altså navn, yrke etc. inspirert av måten det har blitt satt opp på i studentbevisappen. I appen vi utvikler i denne bacheloroppgaven viser vi navn, yrkesstatus, organisasjon/frivillighetssentral og om man er aktiv eller ikke.

```
this.backClicked = function() {
    // Prevents from going back if previous screen if on frontpage
    if (self.router.currentState().id != "frontpage") {
        // Goes back to front page if on profile page, regardless of previous page
        if (self.router.currentState().id == "profile") {
            self.router.go("frontpage");
            for (var i = stateArray.length; i > 1; i--) { // Clears the array
                stateArray.pop();
            }
        }
        // Checks if it should skip a view (used after someone saves an edit)
        else if (self.router.currentState().id == stateArray[stateArray.length - 3]) {
            self.router.go(stateArray[stateArray.length - 4]);
            stateArray.pop(); stateArray.pop();
            stateArray.pop(); stateArray.pop();
        }
        else { // None of the checks match, goes back normally
            self.router.go(stateArray[stateArray.length - 2]);
            stateArray.pop(); stateArray.pop();
        }
    }
}
```

Listing 5.2: Kode for tilbake-knapp



Figur 14: Dette viser studentbevisappen for en gyldig bruker. [18]

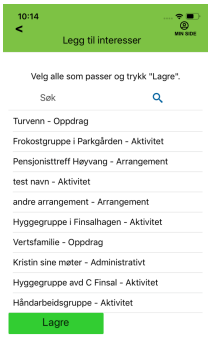


Figur 15: Dette viser ID-kortet for en gyldig bruker i den nye FRIDA-appen (den svarte rundingen er der bilde vil vises).

```
self.validateButtonClicked = function() {
    var elem = document.getElementById('avatarDiv');
    // Animate the element
    oj.AnimationUtils['flipOut'](elem, {'flipTarget': 'children',
                                        'persist': 'all',
                                        'startAngle': '0deg',
                                        'endAngle': '360deg'});
}
```

Listing 5.3: Koden for å vise at det er et ekte ID-kort

Figur 16: Dette viser ”Mine interesser”-skjermen med en X ved hver interesse for å fjerne den som en interesse.



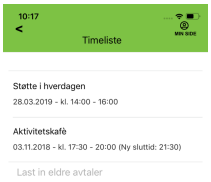
Figur 17: Dette viser interesser som ikke har blitt lagt til i ”Mine interesser” med en lagreknapp for å legge til en eller flere av dem som noe man er interessert i.

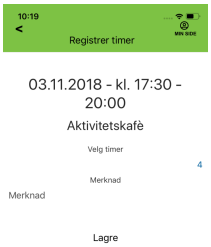
Koden for å vise at det er et ekte ID-kort kan sees i kodesnutt 5.3. Som vi ser her så er det kun to linjer med kode som blir kalt. Den ene får tak i elementet som skal animeres, altså avatarbildet, mens den andre animerer dette elementet. Dette er et fint eksempel på at Oracle JET gjør det veldig enkelt å gjøre visse ting enklere enn det måtte ha blitt gjort med vanlig Javascript eller jQuery da dette ville kreve flere, mer vanskeligere kommandoer.

5.3.6 Mine interesser

På mine interesser så kan man legge til og fjerne interesseområder man har. Med interesser så menes her forskjellige tjenestetyper som man kan lage flere spesifikke tjenester for. Eksempler på dette er ”Besøksvenn”, som betyr at man besøker noen, og ”Avlastning demens”, som betyr at man avlaster for noen med demens, men det kan også være ting som ”Lekse-hjelp”.

Når vi implementerte sidene for ”Mine interesser” så var tanken at man enkelt skal kunne se, fjerne og legge til nye interesser. Derfor la vi til interessene til personen i en liste med en X for å kunne fjerne denne interessen (se figur 16). På ”Legg til nye interesser” så kan personen legge til nye interesser ved å trykke på en eller flere interesser og deretter på ”Lagre”-knappen (se figur 17). Vi har også sortert denne listen slik at kun interesser som både er lagt til av organisasjonen personen er en del av og ikke allerede er lagt til som en interesse av personen selv, vises. I tilegg så vises maks 10 elementer av gangen, og selv om vi henter ned alle elementer med en gang, så må man søke for å se flere elementer enn de 10 første. Når vi søker så gjøres dette lokalt. Dette har blitt bestemt fordi vi ikke ønsker å bruke unødvendig mye datatrafikk, siden hvis vi kun ville hente ned de 10 første elementene og så søke direkte i databasen så ville vi enten måtte la brukeren vite så nøyaktig hva de leter etter at de måtte kunne få maks 10 resultater tilbake, eller søke så mange ganger at det ville blitt en del unødvendig data. Dette får vi løst ved å kun hente alt en gang. Men antallet interesser som blir hentet (sorteringen på organisasjon og person skjer i databasen) er ikke så mye at det blir masse databruk for de fleste organisasjoner av hva vi kan se i testdatabasen.





Figur 18: Dette viser listen med tjenesterFigur 19: Dette viser hvor man kan endsom den frivillige har gjort. Klikk på enre timer og se mer informasjon om hver tjeneste for å se mer info og endre timer.tjeneste.

5.3.7 Endre timer

Hvis noe skjer slik at antallet arbeidstimer rundt en tjeneste avviker fra det som var satt opp først, så kan man gå inn på ”Endre timer”. Her vil man kunne se alle de nyeste tjenestene man har gjort, fra og med dagens dato og bakover i tid. Her kan man endre hvor lenge en hendelse varte i antall timer, mens vi i listen over alle de nyeste tjenestene ser en evt. ny tid som et klokkeslett. I tillegg kan man skrive en merknad ang. endringen av timen (se figur 18 og 19).

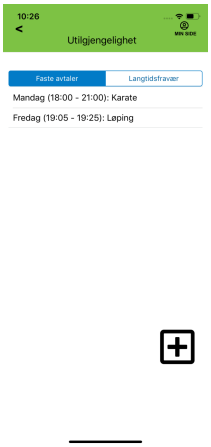
Grunnen til at gruppen har valgt å skrive det som klokkeslett i listen, men at man endrer det som antall timer når man endrer det, er fordi man på den orginale webapplikasjonen endrer det som timer, men at det i visningsøyemed ser bedre ut med et klokkeslett. Da betyr det at man kan endre det slik man er vant med fra webapplikasjonen, men kan enkelt se når noe sluttet som et klokkeslett.

Antallet tjenester kan være så mye at vi har satt en grense på hvor mye vi henter fra databasen om gangen. Denne funksjonaliteten ville gruppen egentlig ha implementert ved å la brukeren av systemet scrolle ned til bunnen av listen, og deretter kalles en funksjon som hentet flere. Men desverre så har Oracle Jet ikke en enkel måte å gjøre dette på. De har en måte å refreshe tabellen slik, men da må man scrolle opp, og det er ikke naturlig i en slik situasjon. Derfor valgte vi å ha en knapp i bunnen som var tilgjengelig for trykk når det eksisterer flere tjenester man kan hente inn. Dette får man info om når man kaller REST-apiet, slik at dette var enkelt å fikse.

5.3.8 Utilgjengelig

På denne siden kan man si at man er utilgjengelig, enten ved å sette en ukentlig hendelse (avtale), eller ved å sette en langtids-engangshendelse (ferie).

Lengtidshendelsene er på minst en dag, mens de ukentlige hendelsene er begrenset til å ikke gå over mer enn en og samme dag. Grunnen til at gruppen ikke har tillat å gå over mer enn en dag er fordi det er skjelden en ukentlig hendelse går over midnatt, og det da som regel er langtidsfravær.



Figur 20: Dette viser utilgjengeliglistenfor den frivilliges avtaler.

Figur 21: Dette viser redigeringen for utilgjengelighetsavtaler.

På figur 20 kan man se en liste over ukentlige avtaler, mens man på figur 21 kan se hvordan man redigerer en avtale. For ferier ser det någelunde likt ut med bare litt forskjellig input.

5.3.9 Tjenester

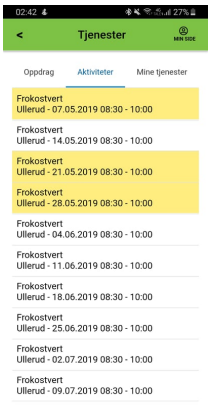
På denne siden finner man lister over tilgjengelige oppdrag og aktiviteter. Om en frivillig har blitt forespurt om å gjøre et oppdrag vil disse listes opp under fanen ”Oppdrag” og den frivillige vil ha muligheten til å akseptere eller avslå oppdraget. Under ”Aktiviteter”, som i figur 22, vil alle aktiviteter som har behov for frivillige bli listet opp, her kan en frivillig melde fra om han/hun er interessert i å hjelpe til eller ikke. Etter at en frivillig har meldt interesse for en aktivitet må en administrator godkjenne/bekrefte at den frivillige skal delta. Under den tredje fanen vil alle tjenestene som den frivillige har påtatt seg eller meldt interesse for bli listet opp, dette er for at det skal være lettere å få en oversikt over kommende hendelser. På alle fanene er tjenestene sortert etter dato slik at den førstkommende tjensten blir vist øverst. Gruppen har også valgt å legge inn en enkel fargekode:

- Tjenester man skal delta på vil være farget i grønt
- Tjenester det er meldt interesse for, men som enda ikke er bekreftet vil være farget i gult
- Tjenester som er avslått vil være farget i rødt.

På figur 23 kan man se et eksempel på informasjonen som vises dersom en frivillig trykker seg inn på en aktivitet, mer informasjon om tjenesten blir vist og den frivillige får også muligheten til å endre på sin status i forhold til aktiviteten.

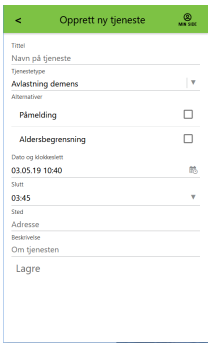
5.3.10 Opprette tjenester

Som administrator har man muligheten til å opprette nye tjenester. Figur 24 viser hvordan siden for å opprette en tjeneste ser ut. Dersom noen av avkrysningsfeltene blir krysset av vil det vises flere felter som må fylles ut, ”Påmelding” vil legge til et felt hvor man bestemmer antall deltakere og ”Aldersbegrensning” vil legge til felter hvor man bestemmer øvre og/eller nedre aldersgrenser.

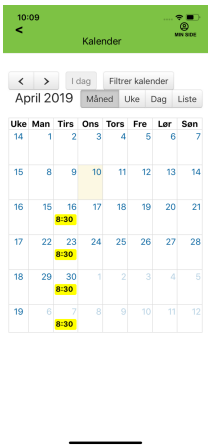


Figur 22: Den nye appens side for tjenester.

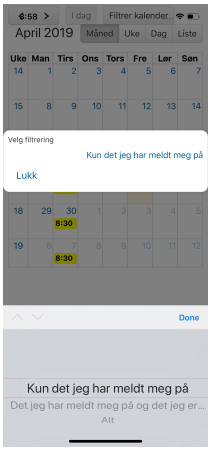
Figur 23: Den nye appens side for informasjon om tjeneste.



Figur 24: Den nye appens side for å opprette tjenester.



Figur 25: Den nye appens frivilligkalender.



Figur 26: Den nye appens frivilligkalender sin filtrering.

5.3.11 Kalender

Frivilliges kalender

Kalenderen viser en frivillig sine tjenester. I tillegg kan også tjenester i en frivillig sine interesseområder og alle tjenester hos deres frivillighetssentral vises om det er ønskelig. Dette gjøres via en filter-knapp.

Tjenestene er fargekodet etter antall frivillige som er påmeldt i forhold til antall frivillige som er ønsket.

På figur 25 kan man se en enkel kalender fra frivilliges synspunkt. Du kan også se på figur 26 hvordan det å endre filtrering ser ut.

Adminkalender

Kalenderen som vises for administratoren viser alle tjenester i den spesifikke frivillighetssentralen admin er en del av. Disse er også fargekodet i forhold til antall frivillige.

i tillegg til kalenderen laget med Fullcalendar-biblioteket kan administratoren se en liste over tjenester det haster å skaffe frivillige til. Denne listen er også fargekodet i forhold til om det er noen frivillige overhode eller ikke.

Denne listen kan også bli sett i Fullcalendar-kalenderen ved å sette filtrering til å vise dette. Filtreringsalternativene til administrator er å vise alle i deres organisasjon og å vise kun de tjenestene det haster å gjøre noe med. Mens listen kun viser for de 10 neste dagene, så viser kalenderen for hele måneden når den viser hele måneden, og for neste uke når kalenderen er i ukesmodus.

På figur 27 så kan man se hvordan en admin kan se kalenderen. Som man ser her så ser den ganske lik ut som frivilligkalenderen unntatt tabben øverst i bildet. På denne figuren ser vi også hvordan kalenderen gjør det når den ikke har plass til å vise alle tjenester. Da viser den nemlig "+x flere", der x er antallet flere tjenester. Hvis man trykker på denne åpnes listen vist i nedre høyre hjørnet av figur 27. Hvis man så trykker på en tjeneste fra admin sin kalender vil popupen vist i figur 28 vises. En lignende popup vil bli vist for frivillig, men ikke med like mye informasjon. ”Se mer”-knappen vil gå til tjenestens egen info-side.

Figur 27: Den nye appens adminkalender.

Figur 28: Den nye appens adminkalender sin informasjon om en tjeneste.

Oppsett av Fullcalendar for frivillig og adminkalender

```
function resetCalendar() {
    $(function() {

        // page is now ready, initialize the calendar...

        self.calendar = $('#calendar').fullCalendar({
            customButtons: {
              filter: {
                text: 'Filtrer kalender',
                id: 'filterButton',
                click: function() {
                  var popup = document.getElementById('filterPopup');
                  popup.position = self.getPosition();
                  popup.open('body');
                }
              }
            },
            aspectRatio: 0.9,
            header: {
              left: 'prev,next today filter',
              center: 'title',
              right: 'month,agendaWeek,agendaDay,listMonth'
            },
            locale: 'nb',
            firstDay: 1,
            dayNamesShort: ['Søn
','Man','Tirs','Ons','Tors','Fre','Lør'],
            dayNames: ['Søndag', 'Mandag', 'Tirsdag', 'Onsdag',
'Torsdag', 'Fredag', 'Lørdag'],
            monthNames: ['Januar', 'Februar', 'Mars', 'April', 'Mai',
'Juni', 'Juli',
            'August', 'September', 'Oktober', 'November', 'Desember'],
            monthNamesShort: ['Jan', 'Feb', 'Mar', 'Apr', 'Mai',
'Jun',
            'Jul', 'Aug', 'Sep', 'Okt', 'Nov', 'Des'],
            weekNumberTitle: "Uke",
            timeFormat: 'H(:mm)', // uppercase H for 24-hour clock
            timezone: 'Europe/Oslo',
            buttonText: {
              today:    'I dag',
              month:    'Måned',
              week:     'Uke',
```

```
39                day:      'Dag',
40                list:     'Liste'
41              },
42              eventLimitText: 'flere',
43              weekNumbers: true,
44              navLinks: true, // can click day/week names to navigate
    views
45              editable: false,
46              eventLimit: true, // allow "more" link when too many
    events
47              slotLabelFormat: "H(mm)",
48              timezone: "Europe/Oslo",
49
50              events: function(start, end, timezone, callback) {
51                self.events([]);          // Initiate/empty events.
52                // Get all events the volunteeer is working on
53                let volunteerUrl = /* URL */;
54                getEvents(volunteerUrl, callback, false, start, end,
    timezone);
55              },
56
57              eventClick: function(calEvent, jsEvent, view) {
58                console.log("ID for event: " + calEvent.eventId);
59                currentEventId = calEvent.eventId;
60                console.log($(this));
61                self.popupTitle(calEvent.title);
62                self.popupPlace(calEvent.place);
63                self.popupContent(calEvent.description);
64                var popup = document.getElementById('popup1');
65                popup.position = self.getPosition();
66                popup.open('body');
67              },
68
69              // Special options for each view:
70              views: {
71                basic: {
72                  // options apply to basicWeek and basicDay views
73                },
74                agenda: {
75                  // options apply to agendaWeek and agendaDay views
76                },
77                week: {
78                  // options apply to basicWeek and agendaWeek views
79                  columnHeaderFormat: "ddd D/M"
80                },
81                day: {
82                  // options apply to basicDay and agendaDay views
83                  columnHeaderFormat: "dddd D/M"
84                }
85              }
86          })
87      });
88  }
```

Listing 5.4: Oppsett av Fullcalendar for frivilligkalender

For å sette opp en kalender med Fullcalendar er det mange innstillinger man kan endre på. Hvordan vi har satt opp kalenderen for frivillige kan sees i kodesnutt 5.4. Oppsettet av administrator er nesten helt likt.

i kodesnutt 5.4 så ser vi både mange innstillinger og noen onclick-funksjoner. På linje 7 til 17 blir filtrer-knappen definert med en funksjon for når knappen trykkes. Her ser vi at det er kjempeenkelt å definere denne. Denne funksjonen bruker en popup definert i HTML-filen for denne siden, og som er en Oracle JET komponent, til å vise valgene man har for filtrering.

I header-delen av definisjonen til kalenderen (linje 19 til linje 23) så settes hvor de for-

skjellige knappene og tekst i kalenderen skal settes i headeren. Det fins også en footer-del om man vil ha noe der, slik som det er fortalt om i del 3.1.7, men dette har vi ikke brukt i denne appen. Når man setter de forskjellige knappene og teksten inn (der knappene ”prev”, ”next” og lignende er predefinerte), så er der de blir delt opp med komma bety at de er helt inntil hverandre, mens et mellomrom betyr at det er noe mellomrom mellom disse delene. ”locale” blir satt til å være norsk, men av en eller annen grunn så ville ikke den norske oversettelsen funke. Dette mest sannsynlig fordi gruppen ikke hadde fått importert denne filen korrekt. Men derfor trengte vi mesteparten av koden som er på linje 25 til linje 48. Hvis vi hadde fått den norske språkpakken ordentlig importert ville vi ikke trengt disse. Attributtet ”firstDay” brukes for å definere hvilken dag som skal være først (der default er søndag, som har id 0, da dette brukes i noen land). Linje 25 til linje 42 er rene oversettelsestekster for at Fullcalendar skal funke.

Funksjonen definert for ”events” på linje 50 til 55 blir kalt hver gang noe skjer som oppdaterer events. En slik oppdatering kan være at neste-knappen trykkes. Her brukes startdatoen og sluttdatoen sendt inn i denne funksjonen til å finne ut av hvilke tjenester som skal vises. Herifra kalles funksjonen getEvents(..). Denne funksjonen får ut tjenestene i tidsrommet som har blitt definert, og ber Fullcalendar om å vise dem.

Deretter ser vi koden for ”eventClick”. Denne funksjonen, som er på linje 57 til 67, definerer hva som skjer når en frivillig trykker på en tjeneste. Her ser vi at koden, i likhet med koden for filtreringsknappen, bruker en forhåndsdefinert popup av typen Oracle JET har som komponent. Denne kan man se viser info gitt av noen variabler. Til slutt så defineres noen deler som er spesifikt for hvert view, som betyr spesifikt for måned, uke, dag, etc.

5.3.12 Lister

I appen er det noen lister som lister opp forskjellige ting for admin. Disse listene er bygd opp slik at de laster ned alle elementer som skal lastes. Denne listen viser deretter bare de 10 første elementene. Deretter kan man enten laste flere elementer om dette eksisterer, eller søke. Denne søkefunksjonen søker lokalt fra det som ble hentet ned da siden ble lastet.

Siden det i noen tilfeller kan være ganske mange elementer som skal lastes, kan dette gå ut over databruken. Vi bestemte oss derimot å la dette være da det i størst mulig grad mest sannsynlig må søkes i disse tilfellene, og for en lik opplevelse av det vi har nå ville blitt mange søk mot backend hvis søket skulle blitt gjennomført der.

I kodesnutt 5.5, så står funksjonen som blir brukt til å søke igjennom arrayet ”array” med søkestrengen ”value”. Deretter blir alle elementer som passer med søkestrengen lagt i et nytt array som, når alle elementer har blitt gått igjennom, returneres. Søkestrengen kan enten starte, stoppe eller være midt i tekststrengen/elementet som sjekkes.

5.3.13 Kamera

Administratorer har tilgang til en kameraknapp, som lar brukeren ta et profilbilde av en frivillig. Denne funksjonen brukes for eksempel når en frivillig har vært på intervju ved en frivillighetssentral. Kamerasiden lar brukeren ta et bilde, og ber så brukeren enten bekrefte bildet eller ta nytt. Om brukeren bekrefter bildet vil det lastes opp i databasen og brukes som den frivilliges profilbilde på blant annet ID kortet.

Det oppstod flere problemer med implementasjonen av denne funksjonaliteten. De fleste kilder på nett snakket kun om hvordan en kunne velge bilder fra mobiltelefonen, men ikke ta dem selv. Da dette ikke var ønsket funksjonalitet, ble det brukt mye tid på å finne en løs-

```
function filterByProperty(array, value){
   var filtered = [];
   for(var i = 0; i < array.length; i++){
       var obj = array[i];
       // Lowercase item, category and search-value so the
       // search is not case-sensitive:
       var str = obj.item.toLowerCase();

       // If value is undefined accept all:
       if (value == undefined || str.indexOf(value.toLowerCase()) !=
   -1) {
         filtered.push(obj)
       }
   }

   return filtered;

 }
```

Listing 5.5: Funksjon for å søke igjennom lister

```
   function openCamera() {

       var srcType = Camera.PictureSourceType.CAMERA;
       var options = setOptions(srcType);

       navigator.camera.getPicture(
         function (imageURI) {
           rawURI = imageURI;
           imgData = '' + imageURI;
           let img = document.getElementById("displayImage");
           img.src = imgData;
         },
         function  (err) {
           //fill out
         },
         {
           quality: 50,
           destinationType: Camera.DestinationType.FILE_URI
         }
       );
     }
```

Listing 5.6: Funksjon for å ta og vise bilde

ning. Omsider viste en kilde til Cordova pluginen Camera. Camera pluginen inneholder kode som tar i bruk kamera på fem forskjellige plattformer, og lager egne funksjoner i Javascript som automatisk bruker riktig kode. Pluginen viste seg fort å ikke fungere på flere områder. I nettleser brukte den foreldet Javascript kode som ikke lenger fungerte i Google Chrome. Firefox aksepterte fremdeles koden, men ga varsler om at koden snart vil slutte å fungere her også. På Android enheter skjedde det heller ingenting, verken når brukeren åpnet kamera eller prøvde å ta et bilde.

Ettersom pluginen stort sett fungerte på Firefox, valgte vi å jobbe videre med den der. Da kunne vi jobbe videre med å sende bildet til databasen og ta det i bruk. Når bildet tas lagres det som en base64 streng, som vist i kodesnutt 5.6. Denne vises i appen, slik at brukeren kan velge om han/hun vil bruke bildet. I databasen er profilbilder lagret som Binary Large Object data, forkortet til BLOB. Ingen i gruppen hadde brukt dette tidligere, og det var vanskelig å få hjelp med dette hos EVRY. Vi fant en funksjon på nett som konverterte bildet fra base64 til

```
$.ajax({
    dataType: "json",
    url: url,
    success: function(person) {
        self.data = {
        // Add name and other things to show to variable self.data
        // person-variable contains parsed json-data.
        };
        self.datasource(self.data); // Add to an observable-variable.
    },
    error: function() { self.rootViewModel.showGeneralError(); }
});
```

Listing 5.7: JQuery sin $.ajax funksjon fra ”Vis ID”-siden

BLOB, og brukte denne før vi sendte all nødvendig informasjon til databasen. Problemet oppstod da databasen kun lagret et tomt BLOB element, selv om elementet i POST-forespørselen inneholdt data. Da dette viste seg å være vanskelig å fikse, endte vi opp med å la kamera funksjonen forbli uferdig, spesielt ettersom vi befant oss i slutten av siste sprint og hadde begrenset tid igjen til utvikling.

5.4 Valg under utvikling

I denne delen blir det beskrevet hvilke valg som har blitt gjort i løpet av utviklingsprosessen ang. implementasjon.

5.4.1 XMLHttpRequest VS $.ajax VS $.getJSON

Gruppen har valgt å hovedsaklig bruke jQuery-funksjonen ”$.ajax” istedenfor den native javascript-funksjonen ”XMLHttpRequest”. Dette var et valg som ble gjort underveis. Siden $.ajax er en jQuery-funksjon så betyr dette at det er en forenkling av native javascript funksjoner. Dette kan sees i kodesnutt 5.7, som er både kortere og enklere å lese enn kodesnutt 5.8, som viser hvordan XMLHttpRequest-funksjonen for det samme ville se ut.

Som man her kan se, så er det mer kode som må skrives for å kunne gjøre akkurat det samme i ”XMLHttpRequest”. Deler av dette er fordi funksjonen som blir kalt av ”onreadystatechange” blir kalt hver gang statusen endres, som for eksempel når den lastes. Merk at dette betyr at man også må sjekke at ”this.readyState” er 4 men forskjellig fra statuskodene som er forskjellig fra success-kodene som kan komme fra responsen for å kunne gi error. Selv om dette i noen tilfeller kan være fint å kunne gjøre, vil man i de fleste tilfeller være fornøyd med å la jQuery definere når det er ”success” og når det er ”error”.

Det ble også en stund, før vi hadde bestemt oss for hvilket ajax kall vi skulle bruke, brukt jQuery-funksjonen ”$.getJSON” i noen GET-kall. Dette er egentlig bare en wrapper av det vi så i kodesnutt 5.7, minus mulighet for å definere error. Grunnen til at denne funksjonen ble endret til å bruke ”$.ajax”, var fordi det senere må settes autorisering på api-kallene. For at man skal kunne sette OAuth-autorisering trengs nemlig headere å bli satt, noe ”$.getJSON” heller ikke støtter.

```
var xhttp = new XMLHttpRequest();
xhttp.open("GET", url, true);
xhttp.send();

xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        // Typical action to be performed when the document is ready:
        var person = JSON.parse(this.responseText);
        self.data = {
        // Add name and other things to show to variable self.data
        // person-variable contains parsed json-data.
        };
        self.datasource(self.data); // Add to an observable-variable.
    }
    else if (this.readyState == 4 && this.status != 200) {
        self.rootViewModel.showGeneralError();
    }
}
```

Listing 5.8: Hvordan XMLHttpRequest-funksjonen ville sett ut for ”Vis ID”-siden

5.5 Forskjeller mellom platformer

5.5.1 Problemer med Cordova for IOS

Cordova, tjenesten Oracle JET bruker for å lage det som ser ut som native-applikasjoner ved å blant annet sette kodebasen inn i et webview i en native app, har hatt noen problemer spesielt med iOS i de nyeste versjonene. Mer spesifikt så har problemet vært noe med at Xcode 10 har en ny kompilator, noe Cordova ikke støtter enda. Det skal ifølge noen gå an å løse, men det tok lang tid før vi klarte å løse det sikkelig. I tillegg mistenkte gruppen at det også var et annet problem i tillegg som gjør at appen heller prøver å åpne html-filene i nettleseren enn i WKWebViewet i appen. Det siste problemet viste seg å la seg løse ved å bare sørge for at det siste som ble bygget med ”ojet”-kommandoen var ”ojet serve ios” eller ”ojet serve ios –device” og ikke ”ojet serve ios –browser”.

For å kunne lage en midlertidig løsning for hovedproblemet for testing så har den av oss i gruppen som både har en iPhone og en Mac kjapt laget en liten app som viser en nettside i et WKWebView. Da kunne vi enkelt teste ut appen på en iOS-enhet ved å kjøre ”ojet serve ios –browser” på en PC eller Mac som har korrekt port åpen om dette er nødvendig og deretter angi korrekt adresse i denne appen. Koden for å lage denne appen kan vi se i kodesnutt 5.9. Som vi kan se her så spør testappen om linken til den kjørbare appen (eller en annen nettside), via en alert, noe som gjør at det kun er WkWebView-et som synes etterpå.

Gruppen klarte etterhvert å få Cordovas kode til å funke ved å blant annet sette build-systemet til ”Legacy”. Da kom det fremdeles error om at iphone ikke ville kjøre, dette ble derimot fikset ved å følge instruksjonen funnet på [19]. Dette tok det veldig lang tid å fikse fordi det var flere problemer på en gang. Dette siste problemet var fordi vi brukte Legacy-systemet på Xcode 10, men dette var ikke en gang nevnt på [20], hvor det står om hvordan vi kunne fikse hovedproblemet.

Derimot funker det nå å bortimot normalt å bygge en kjørende app, så lenge man husker å skrive ”ojet serve ios”, og selv om dette vil gi error, gå inn i XCode og kjøre derifra. Dette blir akkurat som når ”ojet build ios” brukes, som er kommandoen som vil bli brukt for å sende appen til produksjon.

```
1  import UIKit
2  import WebKit
3
4  class ViewController: UIViewController {
5
6      @IBOutlet weak var webview: WKWebView!
7
8      override func viewDidLoad() {
9          super.viewDidLoad()
10     }
11
12     override func viewDidAppear(_ animated: Bool) {
13         let ac = UIAlertController(title: "Which website will you
       open", message: nil, preferredStyle: .alert)
14         ac.addTextField() { (textField) -> Void in
15             textField.placeholder = "URL"
16             textField.text = "https://www.google.com/"
17         }
18
19         let submitAction = UIAlertAction(title: "Open this url",
       style: .default) { [unowned ac] _ in
20             self.webview.load(URLRequest(url: URL(string:
       ac.textFields![0].text!)!))
21         }
22
23         ac.addAction(submitAction)
24
25         present(ac, animated: true)
26     }
27
28
29 }
```

Listing 5.9: Koden som ble brukt til å lage en midlertidig testapp i Swift for iOS

5.5.2 Implementasjon av plugins for Cordova

Som nevnt i 5.3.13 så var det en del problemer med kamera-pluginen til Cordova. Dette førte blant annet til at det under kjøring av appen kunne feile på grunn av problemer med denne pluginen. Det som desverre er dumt med dette er at man må finne løsninger på problemet for hver platform som hadde feil på grunn av pluginen.

5.5.3 Utseende på elementer som for eksempel tekstfelt

Utseendet på elementer som tekstfelt er forskjellig ut i fra hvilken platform som kjøres slik at de ser ut som en native app. Kodemessig gjøres dette av Oracle Jet så lenge deres komponenter brukes.

5.5.4 Appens komponenters plassering i forhold til mobilens synlige komponenter

De fleste telefoner har en form for statusbar og lignende komponenter som alltid vises. Disse komponentene kan være forskjellig plassert og konfigurert på forskjellige platformer som IOS og Android. Et slikt eksempel er at statusbaren på IOS går over WKWebView'et som viser selve appen. Dette gjør at hvis man ser på forsiden i figur 12, som er vist på IOS, så går statusbaren nesten over "Min side"-knappen. Men hvis man gjorde noe med dette kodemessig i appen kunne appen sett rar ut i Android da forholdet mellom statusbaren og WebView'et vil være annerledes. Det går selvfølgelig an å endre noen ting rundt innstillingene i den native appen, men det er småting som kan ta tid å fikse på. Derfor er dette ikke fikset foreløpig.

5.6 Generelle kommentarer rundt koden

Generelt sett så er koden, inkludert variabelnavn og kommentarer skrevet på engelsk mens REST-apiene er skrevet på norsk. Koden er skrevet på engelsk da dette er språket gruppen føler passer best til å kode i. Vi bestemte oss derimot å lage REST-apiene på norsk da databasen var på norsk, altså at den hadde rader som var navngitt på norsk.

Men selv om koden er skrevet på engelsk, så er selvfølgelig all tekst som kan bli skrevet ut på skjermen på norsk. Dette fordi det er en norsk app som er beregnet på et norskt publikum og fordi bacheloroppgaven ikke innebar å ha støtte for flere språk.

6 Distrubusjon

6.1 Testdatabasen og produksjonsmiljø

Testdatabasen som har blitt brukt under denne utviklingsfasen var en egen database gitt til oss av EVRY. Dette gjorde at vi kunne redigere så mye vi ville i databasen da dette ikke ville endre noe i produksjonsdatabasen. Som nevnt i del 3.1.5 så var testdatabasen en Oracle Database med de fordelene det har. Med fordeler menes at de støtter språket PL/SQL for mer avanserte backend-koding direkte i databasen. I tillegg til at Oracle REST Database Services (ORDS), som er nevnt i del 3.1.6, støtter denne databasen.

Resten av produksjonsmiljøet er som nevnt blant annet i 3.1.5 og 3.1.6 SQL Developer og Oracle REST Database Services. SQL Developer er som nevnt tidligere et program som kommuniserer med databasen på samme måte som phpMyAdmin, bare at den blant annet også støtter å lage api-endpoints til ORDS.

Databasen og ORDS-tjeneren som er koblet sammen ligger på EVRY sin utviklingsserver. Dette gjør at vi hele tiden har hatt tilgang på en remote server. Dette gjør at gruppen har kunnet bruke appen også på telefonene helt fra starten av uten å måtte koble til en datamaskin for å få tilgang til api-ene. Dette er unntatt i den perioden vi måtte bruke vår egen fiks for å kunne kjøre på en ekte iOS-enhet, som det står mer om i del 5.5.1, men det var for å få tak i selve appen, da apiet fremdeles lå på EVRY's utviklingsserver.

6.2 Hva må gjøres før appen kan bli satt ut i produksjon

Som nevnt i del 4.3.3 er det enda ikke lagt på autorisering på api'et fordi EVRY fikk problemer da de prøvde å hjelpe oss med dette på serversiden. Derfor må dette bli gjort. I tillegg så skal det som nevnt tidligere ikke bli brukt innloggingsfunksjonaliteten som vi la på, men endres til den de bruker. Det er også noen mindre ting som ble oppdaget mot slutten av bacheloroppgaven som gruppen ikke rakk å endre på. Dette innebærer for eksempel, som man kan se i 2.9 at "Tjenester"-tabellen både inneholder en "merknad"-kolonne og en "beskrivelse"-kolonne, mens vi kun har brukt "merknad", mens vi burde lagt opp til å bruke begge to. Men vi fant også noen andre småting som vi rakk å fikse på.

Deretter så kan vi bygge et program for hver platform, altså en .apk-fil for Android og .ipa-fil for iOS. Dette gjøres ved å kalle "ojet build android" og "ojet build ios". Deretter må det sendes inn en forespørsel til Google og Apple for publisering. Mens Google gjør sjekk på appen automatisk så går Apple gjennom alle nye apper manuelt.

7 Testing og tilbakemeldinger fra brukere

Gruppen har i løpet av utviklingsprosessen fått gjennomført en brukertest hos Hamar frivillighetssentral i tillegg til flere mindre demovisninger av appen til Evry for tilbakemeldinger underveis i Scrum-prosessen.

7.1 Testing hos Hamar frivillighetssentral

Hamar frivillighetssentral er en av frivillighetsorganisasjonene som benytter seg av FRIDA. Gjennom våre kontaktpersoner i EVRY fikk vi gjennomført en test med to som er frivillige i Hamar, tilbakemeldingene fra disse testpersonene var ekstra verdifulle ettersom de var brukere av det gamle systemet. Testen ble gjennomført av to personer, en mann på 28 år med høye digitale ferdigheter (referet til som testperson 1) og en dame på 51 år med gjennomsnittlige digitale ferdigheter (referert til som testperson 2).

7.1.1 Mål med testen

I løpet av utviklingen har brukervennlighet hatt mye fokus, et av hovedmålene med testen var derfor å lære mer om hvor enkelt testpersonene syntes det var å navigere rundt om i appen. I tillegg ønsket gruppen å vite om testpersonene syntes noe burde gjøres annerledes eller burde legges til.

7.1.2 Fremgangsmåte

Innledning

Før testingen kan begynne er det viktig å informere testpersonene om hvordan dataen vi samler inn blir behandlet og hva formålet med testingen er. Følgende informasjon ble gitt til testpersonene:

1. Denne testen skal kartlegge brukervennligheten i den nye FRIDA appen.
2. Det er appen som testes, ikke testpersonen. Hvis noe er vanskelig så er det appen sin feil.
3. All data vi samler fra testen vil lagres anonymt, og vil senere bli brukt i den endelige rapporten.
4. Testpersonen må gjerne kommentere sine tanker underveis.
5. Vi skal gi en rekke med små oppgaver/hypotetiske scenarioer testpersonen kan finne seg i. Testpersonen løser så "oppgaven" slik han/hun mener den skal løses.
6. Testen avsluttes med noen spørsmål om brukervennligheten og funksjonaliteten i applikasjonen

Apptest

Deretter gjennomførte gruppen testen etter dette mønsteret:

1. Logg inn med angitt brukernavn og passord
2. Scenarioer:
    1. Testpersonen har fått ny jobb og med den følger det en ny email. Oppdater til den

nye emailen i FRIDA appen.
2. Testpersonen vil gjerne kunne motta varsler på den nye emailen. Aktiver email-varsler i FRIDA appen.
3. Testperson har foreløpig ingen interesser, og får dermed ingen tilbud om oppdrag. Fiks dette ved å bruke FRIDA appen til å legge til nye interesser.
4. Testpersonen har tidligere deltatt som frivillig på en "Aktivitetskafé", men den varte en time lenger enn planlagt. Bruk FRIDA appen til å justere lengden på aktiviteten.
5. Testpersonen har begynt på et kurs i karate, som har trening hver mandag kl seks til åtte. Bruk FRIDA appen til å vise at du ikke er tilgjengelig da.
6. Karatekurset skal på tur til Japan. Dere reiser 20. mars og er tilbake 1. april. Registrer at du er borte i denne perioden i FRIDA appen.
7. Alle er motiverte etter turen til Japan, og ønsker derfor å trene mer. Treningen på mandag starter nå en time tidligere. Bruk FRIDA appen til å oppdatere dette.
8. Testpersonen har fått et oppdrag om å hjelpe en bruker. Bruk FRIDA appen til å akseptere oppdraget om "Støtte i hverdagen"
9. Det trengs mange frokostverter fremover i tiden, og testpersonen tenker å hjelpe til. Meld interesse som frokostvert med FRIDA appen.
10. Bruk kalenderen til å se hva testpersonen gjorde i desember i 2018. Endre filteret til "Alt" for å se alt som skjedde.

Avsluttende spørsmål

Til slutt spurte vi disse spørsmålene:

1. Var det enkelt å navigere seg i appen? Lå funksjonaliteten der testpersonen forventet det skulle være?
2. Er det noe testpersonen syntes bør gjøres annerledes?
3. Noe som manglet? (Funksjoner, utseende, etc)
4. Andre kommentarer?

Andre tanker for gjennomføringen

Under planleggingen tenkte gruppen også på hva vi måtte få tydelig frem og hvilket tankesett vi måtte ha under utførelsen. Dette er listet opp under:

- Ta tida på hvor lang tid brukeren bruker på de forskjellige oppgavene
- Få bruker til å kommentere tankegang
- Hva vil vi ha svar på? Navigering, funksjonalitet, brukervennlighet, hvor god er den nye i forhold til den gamle
- Huske å informere bruker om hele prosessen
- Anonymitet er viktig
- Lage underskriftsskjema
- Mange små vs noen store omfattende oppgaver
- Kvalitativt intervju om hvordan brukeren opplever applikasjonen og om noe manglet
- Vær tydelige på at vi er åpne for kritikk og er ikke ute etter å dømme brukeren

7.1.3 Hva som ble forandret fra planen når testene ble gjennomført

Selv om vi først tenkte på å ta tida på hvor lang tid brukeren brukte syntes vi under testene at dette ikke var nødvendig da det ikke var avgjørende hvor lang tid som ble brukt, men heller

at det var intuitivt.

I tillegg så hadde gruppen problemer med å gjennomføre testen da vi måtte utsette testen en uke på grunn av at EVRY byttet servere samme dag som vi skulle teste. Mer om dette står i del 4.4.1. På grunn av dette kunne den ene testpersonen vi egentlig skulle ha møte med ikke møte opp, og på grunn av noen misforståelser mellom lederen og den andre testpersonen vi skulle teste på så møtte ikke denne opp heller. Derfor måtte vi finne noen blant de som var tilstede der på dette tidspunktet, som er de som refereres til som testperson 1 og testperson 2.

7.1.4 Resultater

Ingen av testpersonene hadde noen store problemer under gjennomføringen av oppgavene, begge kom frem til at det var enkelt og intuitivt å navigere seg gjennom appen. Mens de avlsuttende spørsmålene ble stilt fikk testpersonene muligheten til å gå gjennom applikasjonen en gang til for å kunne skaffe seg et mer fullstendigt bilde.

Testperson 1 merket seg at det svært sjeldent kom bekreftelser etter at endringer ble gjort, noe som kunne føre til at han ble usikker og følte at han måtte dobbeltsjekke om endringene faktisk ble lagret. Noe annet han nevnte var at selv om det var intuitivt for han å trykke på profilikonet for å komme til ”Min Profil”-siden ville det kanskje være vanskeligere å forstå for eventuelle brukere med mindre digital erfaring.

Testperson 2 satt stor pris på at knappene i applikasjonen var store og enkle å trykke på, utenom på ”Mine interesser” hvor knappene ble litt for små og vanskelige å treffe. Hun savnet fargekoder på tjenestene slik at man enkelt kunne sjekket statusen på de ulike tjenestene uten å måtte trykke seg inn på de og eventuelt en egen liste som kun viste de tjenestene man hadde vist interesse for.

8 Diskusjon

8.1 Tanker rundt valg av programmeringsspråk og rammeverk

Som nevnt i sammenligningen av rammeverk i del 3.1.1, så var en av grunnene til at vi valgte å gå for Oracle JET som rammeverk, at vi fra EVRY fikk en Oracle-database, og håpet at det var en snarvei for å koble til denne siden begge var fra Oracle. Etter at vi startet å bruke Oracle JET så fant vi ut at dette ikke var tilfelle. Istedenfor måtte vi bruke tjenesten ORDS for å lage REST apier, som det står mer om i 3.1.6.

Siden det ikke var noen spesiell kobling mellom Oracle JET og andre programvarer fra Oracle, så kunne gruppen vår ha valgt et annet rammeverk. Men selv om det gruppen forventet var i Oracle JET ikke var der, så var det fremdeles gunstig å bruke Oracle JET istedenfor et annet rammeverk fordi den kommer med mange komponenter, og man kan legge enkelt til flere om man vil det, som for eksempel kalender-biblioteket Fullcalendar som det ble skrevet om i del 3.1.7.

Selv om vi kunne tatt en bedre titt på koblingen mellom Oracle JET og databasen så er det sannsynlig at vi ville ha valgt Oracle JET uansett. Dette fordi EVRY har folk som allerede kunne rammeverket og kunne svare på spørsmål vi hadde. Hadde vi valgt et annet rammeverk ville vi kun ha støtte fra informasjon funnet på nett.

8.2 Endringer i omfang av oppgaven

Da gruppen tok på seg oppgaven var omfanget veldig udefinert, informasjonen som var gitt sa at EVRY ønsket seg en mobilapplikasjon som skulle være en del av deres FRIDA prosjekt. Før utviklingen begynte var det derfor viktig at vi hadde et planleggingsmøte med EVRY hvor vi kunne definere oppgaven tydeligere, resultatet av dette møtet var tre hovedoppgaver med prioritering i følgende rekkefølge:

1. En mobilapplikasjon for frivillige
2. En webapplikasjon med statistikk for frivillighetsorganisasjoner
3. Utvidelse av mobilapplikasjonen til å inkludere administratorfunksjonalitet

Den første oppgaven ble markert som obligatorisk mens nummer 2 og 3 var bonusoppgaver som kunne gjennomføres dersom det ble tid til det. Gruppen så på oppgavene som overkommelige og hadde som mål å klare minst to av de tre oppgavene, men valgte å kun fokusere på den første oppgaven i starten ettersom denne var den største og viktigste av dem.

Da den første oppgaven var nære til å være fullført hadde omkring 60% av tiden som gruppen hadde satt av til utvikling blitt brukt, men gruppen hadde enda ikke begynt på noen av de sekundære oppgavene og det ble derfor arrangert et nytt planleggingsmøte med EVRY for å bestemme fremtiden for utviklingen. Den endelige beslutningen ble å fortsette med den tredje oppgaven som i utgangspunktet hadde lavest prioritet. Grunnlaget for denne avgjørelsen var at den andre oppgaven som var en webapplikasjon praktisk sett kunne vært et helt separat og uavhengig prosjekt, istedenfor å begynne helt fra bunnen av på et nytt produkt var det naturlige valget derfor å fokusere videre på mobilapplikasjonen og utvide

funksjonaliteten til administratorroller i systemet.

Selv om gruppen valgte den tredje oppgaven, som sekundær oppgave, for å slippe å begynne fra bunnen av i et helt nytt prosjekt ble overgangen mellom oppgavene allikevel opplevd som en ny start. Ettersom gruppen tok avgjørelsen om hvilken oppgave som skulle utføres etter oppgave 1 så sent i utviklingsfasen ble resultatet at oppgave 1 fikk alt av fokus inntil den var fullført og først når oppgave 1 var helt ferdig ble fokuset flyttet over til oppgave 3. En konsekvens av denne typen overgang var at utviklingstempoet ble trappet ned når gruppen, mot slutten av oppgave 1, begynte å bruke mer tid på finpussing av funksjonaliteten.

Noe gruppen kunne ha gjort annerledes var å ta avgjørelsen om sekundær oppgave tidligere og dermed kunne integrere funksjonalitet fra oppgave 3 synkront som at gruppen gjorde seg ferdig med oppgave 1. Gruppen fikk mye framgang på oppgave 3, men når utviklingsfasen var ferdig manglet det fortsatt litt for at gruppen kunne levere den som en fullstendig og ferdig oppgave. En smidigere overgang mellom oppgavene kunne bidratt til at utviklingstempoet ikke ville blitt trappet ned og gitt mer tid for å bli helt ferdig med funksjonaliteten til oppgave 3.

8.3 Hva gruppen har lært

Gruppen har jobbet med et helt nytt rammeverk gruppen ikke var kjent med fra før av. Dette har betydd at gruppen har brukt noe tid på å lære oss dette. Gruppen har også brukt flere andre verktøy som gruppen ikke har brukt tidligere slik som ORDS og SQL Developer. Dette betyr at vi har kommet ut fra bacheloroppgaven med et større sett med kunnskap rundt verktøy som kan brukes.

Gruppen har også fått hands-on erfaring ved bruk av Scrum. Dette betyr at gruppen er bedre forbredt i å gå inn i et slikt prosjekt senere, for eksempel i jobbsammenheng.

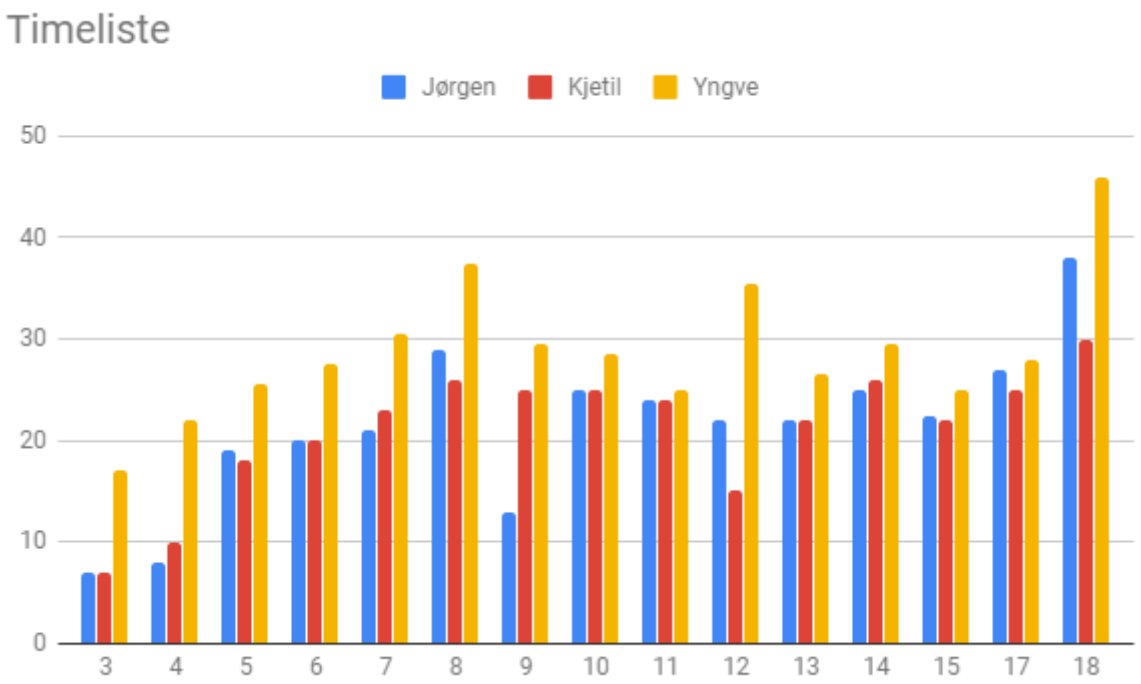
Siden gruppen har jobbet med et stort firma som EVRY og med de ressursene de har, så har vi også fått en innsikt i hvordan det er å jobbe i en så stor bedrift. Dette er fordi vi kan se både hvordan avdelingen vi jobbet med har arbeidet og hvordan bedriften fungerer generelt.

8.4 Evaulering av gruppens arbeid

I prosjektplanen ble gruppen enige om at alle medlemmene skulle jobbe omtrent 25 timer i uken. Etter de første ukene, hvor det ble jobbet mindre da oppgaven ikke var ferdig definert, gikk dette stort sett fint. Alle gruppens medlemmer jobbet sammen hver arbeidsdag gjennom prosjektet, med unntak av enkelte sykdomsdager, spesielt i uke ni og tolv. Gruppen lagde ikke regler for timeregistrering på forhånd. Timene kan derfor variere fra person til person, ettersom medlemmene kan ha notert på ulike måter. Dette vil å så fall bare stå for små unøyaktigheter, og tallene i figur 29 er stort sett riktige.

8.5 Kritikk av oppgaven og prosessen

Da vi fikk oppgaven fra EVRY var den som nevnt i 8.2 veldig udefinert. Men det stod blant annet oppramset verktøy og tjenester som hvilken server FRIDA kjørte på og hva nettsiden og dagens app er laget i, altså Oracle APEX, som om vi skulle bruke disse i kodingen. Men da oppgaven ikke omfattet bruk av serveren da det eneste i backenden vi gjorde var via Oracle REST Database Services (ORDS), så må gruppen si at det gjorde at oppgaven ble litt misvisende beskrevet.



Figur 29: Oversikt over timer jobbet per uke

Vi må også kritisere det faktum at EVRY, som nevnt i 4.4.1, overførte og slo av den første utviklingsserveren samme dag som vi egentlig hadde tenkt til å teste frivilligdelen av appen. Dette betydde som sagt at mange dager gikk bort til å gjennopprette og fikse api'et. I tillegg til at vi måtte vente en uke med å teste hos Hamar frivillighetssentral.

Gruppen vil også nevne at deler av databasen gitt av EVRY ikke var normalisert. Med dette mener vi hovedsaklig at noen av tabellene faktisk inneholdt informasjon som også var gitt av andre deler av databasen.

Ellers så vil gruppen pressisere at sammarbeidet med EVRY har vært utmerket. De var også helt klare på at det var de som hadde skylden da vi ble nødt til å flytte testen en uke fremover.

Gruppen vil ellers også nevne at vi også kunne ha gjort noen ting annerledes selv. Dette inkluderer at det tok en stund før gruppen bestemte en spesiell programmeringsmåte, selv om vi bestemte språk og lignende någelunde kjapt. Dette inkluderer også hvilken ajax-kommando vi skulle bruke, men dette ble som sagt bestemt etterhvert. Selv om gruppen har prøvd å fikse på slike ting så har vi desverre ikke hatt tid til å fikse på alle slike småting, da det er mer kodemessig forskjellige funksjoner, men fungerer likt generelt.

8.6 Brukertesting

Som nevnt i 1.2 har gruppen under utviklingen hatt et fokus for at appen skal være enkel og rask å bruke for en veldig bred målgruppe. Alle medlemmene av gruppen er IT-studenter som har vokst opp med teknologi rundt oss, og dermed blitt veldig hendige med ting som

smarttelefoner. Et problem som kan oppstå som resultat av dette er at det kan være enkelt å legge inn funksjonalitet som gruppen mener er selvforklarende, men som personer som er mindre teknisk avanserte sliter med å forstå. For å oppdage slike feller er det derfor viktig å gjennomføre gode brukertester med aktuelle brukere og sjekke om all funksjonaliteten oppfattes intuitivt av dem.

I løpet av utviklingsprosessen gjennomførte gruppen én reell brukertest som ble utført av to personer. Selv om vi fikk flere gode tilbakemeldinger etter brukertesten var det kun to personer og det er mulig at det er noen ting som også disse har oversett. En av forsøkspersonene prøvde å sette seg selv i skoene til en bruker med lave tekniske ferdigheter, men utenom dette har applikasjonen kun blitt testet av personer med middels til høye tekniske ferdigheter. Testen som ble gjennomført hadde stor betydning for gruppen ettersom testpersonene var brukere av FRIDA og var kjent med den gamle applikasjonen, men for å få et bredere grunnlag bak testingen burde gruppen sannsynligvis gjennomført flere tester. Testing av medstudenter kunne også resultert i gode og mer tekniske tilbakemeldinger samtidig som at det ville vært enklere å organisere.

På bakgrunn av tilbakemeldingene etter brukertesten valgte gruppen å gjennomføre noen endringer i applikasjonen. Den største endringen som ble gjennomført var på siden som omhandler tjenester, se 5.3.9. Originalt var det kun to faner som skilte de to ulike tjenestenetypene fra hverandre, men fra testresultatene 7.1.4 kom det frem et ønske om en ekstra fane for de tjenestene brukeren hadde vist interesse for. En annen, litt mer subtil endring omhandlet knappen som tar brukeren til ”Min Profil”-siden. Denne knappen er veldig sentral i applikasjonen fordi den ligger i headeren, 5.3.4, som vises på nesten alle sidene i applikasjonen. Før testingen var denne knappen kun et ikon, men dette kunne blitt et usikkerhetsmoment for brukere som ikke er vant til at et slikt ikon vil føre til en profilside. Gruppen valgte derfor etter testingen å legge inn teksten "Min side"under ikonet for å understreke funksjonaliteten.

8.7 Videre arbeid

I diskusjonen om oppgavens omfang 8.2 kom det frem at gruppen ble ferdig med hovedoppgaven som var å lage en mobilapplikasjon for frivillige. Det finnes allikevel noen oppgaver som kan gjøres for å forbedre applikasjonen enda et hakk videre. En av oppgavene som var oppe til vurdering var å legge til mer brukerinformasjon på profilsiden. For øyeblikket er det kun navn, epost, mobilnummer, interesser og varslingsinnstillinger som er tilgjengelige for en bruker gjennom applikasjonen, men det finnes mange andre data som er lagret om en bruker i databasen. For gruppen sin del hadde ikke denne oppgaven høy prioritet, alle brukerdataene er imidlertidig tilgjengelige gjennom den nåværende webapplikasjonen.

Når det gjelder administratordelen av applikasjonen er det noe funksjonalitet som fortsatt mangler. Etter at en administrator har opprettet en ny tjeneste er det ønskelig at han/hun også har muligheten til å sende forespørsler til frivillige og be dem ta på seg tjenesten. Kamerafunksjonen som skal la en administrator ta bilde av en frivillig og sette dette bildet som den frivillige sitt profilbilde ble heller ikke helt ferdig, det står mer om dette i 5.3.13. Dersom man ser på forsidebildet for administratorbrukere, 5.3.3, er det mulig å se en knapp for en funksjonalitet som heter ”Send Melding”. Denne funksjonen skulle være en varslingsfunksjon hvor en administrator enkelt og fort kunne sende ut en felles melding til en gruppe frivillige, basert på interesser, deltakere på arrangement eller lignende, og varsle om viktige hendelser.

I forhold til hele FRIDA prosjektet er det også noen brukere som ikke har blitt inkludert

i denne oppgaven i det hele tatt. Tjenestemottakere og deres pårørende er også viktige personer i systemet. For EVRY kan det være mulig å sette opp en bacheloroppgave som gir disse brukerne en plass i applikasjonen også. En tjenestemottaker ønsker kanskje muligheten til å be en administrator opprette en ny tjeneste eller å melde seg på arrangementer som deltaker. Når en frivillig eventuelt har tatt på seg et oppdrag vil gjerne tjenestemottakeren vite noe informasjon om den frivillige som skal komme og hjelpe til. En pårørende ønsker muligens også å kunne be om tjenester for en tjenestemottaker og ha muligheten til å sjekke statusen til tjenestene som omhandler den pårørende.

9 Konklusjon

Gruppen har nå brukt et halvt år på å ferdigstille dette prosjektet. I løpet av dette semesteret har vi lært nye verktøy og fått hands-on erfaring med Scrum og håndtering av klienter og oppdragsgiver. Håndteringen av klienter fikk vi fra testingen hos Hamar frivillighetssentral da vi fikk masse ønsker fra dem.

Det at EVRY har satt folk på saken til å gå igjennom koden og se om de skal bruke appen videre har gledet oss da dette vil gjøre at det vi har jobbet med kan komme ut i produksjon. Dette betyr at vår bacheloroppgave ikke bare vil bli en teoretisk oppgave hvis EVRY bestemmer seg for å utvikle den videre.

Gruppen har også brukt tid på å vurdere språk og rammeverk, og som nevnt så ble det, selv ikke etter gode vurderinger, klart å bli laget en full oversikt over hva som gikk an og ikke gikk an. Dette betyr at selv om man prøver så godt man kan ikke klarer å få alt korrekt på presset tid.

Bibliografi

[1] Gokhan Goksu. Implementing oauth in oracle jet applications. https://blogs.oracle.com/imc/implementing-oauth-in-oracle-jet-applications, 2017.

[2] Oracle-base. Oracle rest data services (ords) : Authentication. https://oracle-base.com/articles/misc/oracle-rest-data-services-ords-authentication, 2018.

[3] jbo5112. Htmlspecialchars equivalent in javascript. https://stackoverflow.com/questions/1787322/htmlspecialchars-equivalent-in-javascript, 2017.

[4] Simon McCallum. Bachelor thesis template (ntnu). https://github.com/COPCSE-NTNU/bachelor-thesis-NTNU, 2019.

[5] EVRY. Evry frida info med bilder. https://apps.evry.no/frida.html, 2019.

[6] Google Developers. Android studio. https://developer.android.com/studio, 2019.

[7] Facebook. React native. https://facebook.github.io/react-native/, 2019.

[8] Oracle. Delivering restful api services for your oracle database. https://www.oracle.com/database/technologies/appdev/rest.html, 2019.

[9] Fusio. The fusio-project. https://fusio-project.org, 2019.

[10] Oracle. The oracle jet architecture. https://docs.oracle.com/middleware/jet410/jet/developer/GUID-293CB342-196F-4FC3-AE69-D1226A025FBB.htm#JETDG113, 2019.

[11] Wikipedia contributors. Model–view–viewmodel. https://en.wikipedia.org/wiki/Model-view-viewmodel, 2019.

[12] Justinmind. Justinmind. https://www.justinmind.com/, 2019.

[13] Difi. Universell utforming: Wcag 2.0-standarden. https://uu.difi.no/krav-og-regelverk/wcag-20-standarden, 2019.

[14] Oracle. Oracle jet and accessibillity. https://docs.oracle.com/en/middleware/jet/6.1/develop/oracle-jet-and-accessibility.html#GUID-9E3452C1-2A85-4700-83B1-B266F348C7E5, 2019.

[15] Techopedia. What is brute force attack? https://www.techopedia.com/definition/18091/brute-force-attack, 2019.

[16] Jonathan. Oauth 2.0 authorization header. https://stackoverflow.com/questions/11068892/oauth-2-0-authorization-header, 2013.

[17] Oracle. Intermitent ora-00604: and ora-01031: Insufficient privileges, using restful services (doc id 2439386.1) (ords error). https://support.oracle.com/knowledge/Oracle%20Cloud/2439386_1.html, 2019.

[18] Felles studentsystem. Studentbevisappen. https://www.fellesstudentsystem.no/studentbevis/, 2017.

[19] Apache. Xcode 9 error: "iphone has denied the launch request". https://stackoverflow.com/questions/45421179/xcode-9-error-iphone-has-denied-the-launch-request, 2018.

[20] Apache. Cordova: Status of xcode 10 support. https://github.com/apache/cordova-ios/issues/407, 2019.

A Prosjektavtale

NTNU

Norges teknisk-naturvitenskapelige universitet

Vår dato Vår referanse

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

EVRY, Brumunddal (oppdragsgiver), og

Jørgen Jarnes

YNGVE HARALD TJELGUM HESTEM

KJETIL HELGAS (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 10.01.19 til 8.06.19.

Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Tom Røise

Oppdragsgivers kontaktperson (navn): Merete Myren

Student(er) (signatur): _____ dato 15.01.19

_____ dato 18.01.19

Kjetil Helgøs dato 18.01.19

_____ dato _____

Oppdragsgiver (signatur): Johnny Bjørnstad dato 18.01.19

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato 23/1/19

B Prosjektplan

Prosjektplan

Kjetil Helgås, Yngve Hestem og Jørgen Jærnes

1 - Mål og rammer

1.1 - Bakgrunn

Det finnes mange frivillige organisasjoner i Norge som organiserer aktiviteter, arrangementer og oppdrag for tjenestemottakere. For mange av disse organisasjonene skjer registrering manuelt, noe som betyr at arrangører selv må ta kontakt med alle som skal delta, både frivillige og tjenestemottakere. Evry har et prosjekt som heter FRIDA (Frivilligsentralenes Data Administrasjonsverktøy) som har som mål å effektivisere hele prosessen og gjøre den enklere for arrangører, frivillige og tjenestemottakere. Dette skal skje ved å utvikle en nettside og en mobilapplikasjon som automatiserer flere av de tidkrevende, manuelle oppgavene.

1.2 - Prosjektmål

- Evry ønsker en ny app til mobil for sin tjeneste for å gjøre det lettere for brukere å bruke tjenesten.
- Evry ønsker i første omgang en app som kan brukes av tjenestemottakere, pårørende og frivillige, men som senere kan bli utvidet til å inkludere administrasjon.
- Applikasjonen skal inneholde funksjoner som blant annet en integrert aktivitetskalender, mulighet for oppdatering av interesse og integrasjon mot sosiale medier.
- Dersom gruppen blir ferdig med appen, ønsker Evry også en nettside med mulighet for å vise statistikk.

1.3 - Rammer

Oppgaven innebærer at gruppen skal bruke en Oracle-database tildelt av oppdragsgiver, men kan ellers velge utviklingsmiljø, rammeverk og språk selv. Databasen inneholder data som kan brukes under utviklingen, og kan også modifiseres dersom det skulle være nødvendig.

Dagens applikasjon kjører i dag som en nettside på en Apache Tomcat-server. Nettsiden tilbyr mobilvisning, som viser innholdet slik det vil fremstå på mobil. Denne visningen inneholder i dag kun enkle informasjonssider og begrenset funksjonalitet. Ettersom mobilvisningen fungerer som en nettside, kan den optimaliseres til å fungere bedre som en mobilapplikasjon.

2 - Omfang

2.1 - Fagområde

FRIDA innebærer at frivillige enkelt kan hjelpe personer med hverdagslige oppgaver de ikke er i stand til å gjøre selv. En slik oppgave kan blant annet innebære følgetjenester, for eksempel hjelpe en person med å dra til et arrangement eller til en legetime. Både tjenestemottaker og pårørende kan be om tjenester for mottakeren.

Liknende tjenester har også vært tilgjengelige før FRIDA, men var vanskeligere å organisere ettersom mottaker måtte sende en forespørsel til frivillighetssentralen, som deretter ringte rundt til frivillige som hadde satt seg opp som interessert i den tjenesten det gjaldt. FRIDA optimaliserer dette systemet ved at all kommunikasjon til både frivillige og mottakere skjer automatisk.

2.2 - Avgrensning

FRIDA er bygget for frivillighetsorganisasjoner som organiserer hverdagslig hjelp. Systemet er dermed ikke ment til bruk av organisasjoner som driver annet frivillighetsarbeid, derav blant annet Røde Kors som driver med søk og redning og Redd Barna som driver med veldedighet. FRIDA har ikke som mål å hjelpe slike organisasjoner, da de trenger andre typer tjenester enn hva systemet tilbyr.

2.3 - Oppgavebeskrivelse

Oppgaven er å utvikle en ny versjon av mobilapplikasjonen til FRIDA, som kan tas i bruk av frivillige, tjenestemottakere og pårørende. Den nåværende applikasjonen lar brukeren se fremtidige arrangementer i sin personlige kalender, i tillegg til å la brukeren se og svare på forespørsler, se og endre tjenester han/hun er interessert i, registrere timer og endre passord. Appen kan også brukes til å vise frem et ID kort som viser hvem brukeren er og hvilken rolle han/hun har. Den nye applikasjonen skal inkludere alle disse funksjonene, i tillegg til å la brukeren se og endre sine personopplysninger, se og endre interesser og registrere fravær.

Dersom den nye FRIDA appen står ferdig tidlig i prosjektet, er andreprioritet å lage en nettside som benytter REST-apier til å vise frem statistikk for de forskjellige organisasjonene. Denne nettsiden skal hente data som allerede ligger lagret i databasen til å vise blant annet hvor mange som har logget på den siste måneden og hvor mange SMSer som har blitt sendt det siste året. Dette vil også hjelpe frivillighetssentralene å holde oversikt over utgifter relatert SMS og email.

Skulle begge de tidligere nevnte oppgavene stå ferdige før prosjektet er ferdig, skal det lages et administrasjonspanel til den tidligere beskrevne appen. Her vil administratorer

kunne registrere tjenester, som treff og aktiviteter, få fremvist en kalender med kommunale tjenester de har ansvar for og kunne integrere denne kalenderen i sin personlige kalender.

3 - Prosjektorganisering

3.1 - Ansvarsforhold og roller

I tillegg til at alle skal være med å utvikle produktet har vi disse oppgavene:

- Leder: Kjetil Helgås - Skal ha hovedansvaret for prosjektet på vår side og vil passe på at vi holder tidsplanen.
- Sekretær: Jørgen Jærnes - Skal skrive ned notater under møter.
- Kontaktperson: Yngve Hestem - Skal holde kontakten mellom gruppen og veileder og andre når dette er nødvendig.

3.2 - Rutiner og regler i gruppa

Ved å delta i prosjektet har alle medlemmer godkjent reglene og er pliktige til å følge dem.

1. Alle uenigheter i gruppen løses i første omgang med avstemning innad i gruppen. Dersom gruppen fremdeles ikke kommer til enighet, benyttes veileder og/eller arbeidsgiver til å finne en løsning.
2. Alle medlemmer av gruppen er pliktige til å jobbe minst 25 timer i uken. Om dette ikke er mulig avtales det innad i gruppen.
3. Alle medlemmer av gruppen er pliktige til å melde ifra dersom de ikke kan møte til avtalt tidspunkt.
4. Utgifter som følge av bacheloroppgaven deles likt mellom alle medlemmer i gruppen. Gruppen må gi samtykke før felles utgifter.
5. Alle medlemmer av gruppen kan signere for gruppen, såfremt minst en annen medlemmer av gruppen gir samtykke.
6. Ved gjentatte og/eller alvorlige brudd på gruppereglene vil det innkalles til møte med veileder for å finne en løsning på dette.

4 - Planlegging, oppfølging og rapportering

4.1 Hovedinndeling av prosjektet

Hele prosjektet har blitt delt opp i tre hovedfaser: planlegging, utvikling og rapportskriving. Planleggingsfasen blir brukt for å få et klarere syn på oppgaven og å ta avgjørelser om utviklingsmetode og utviklingsplattform. Utviklingsfasen er den største fasen og ettersom gruppen har valgt å benytte seg av Scrum er denne igjen delt opp i mindre sprinter. Hver sprint varer i to uker. Den første arbeidsdagen hver uke skal gruppen jobbe hos oppdragsgiveren så det blir derfor naturlig at sprintene starter med et sprint review sammen med oppdragsgiver og deretter et sprint planning møte. På denne måten har gruppen også

muligheten til å få hjelp fra oppdragsgiveren under de første sprint planning møtene og få et litt tydeligere innblikk i hvordan et slikt møte faktisk foregår. Den siste fasen kommer til å ha en overlapping med utviklingsfasen ettersom rapportskriving er noe som bør gjøres synkront med andre oppgaver gjennom hele prosjektet. Gruppen har et mål om at produktet skal være ferdig til 1. mai slik at rapporten kan få fullt fokus i en liten periode.

4.2 - Plan for statusmøter og beslutningspunkter i perioden

Gruppen skal primært jobbe fra tirsdag til fredag hver uke, men kan modifisere dette dersom det skulle være nødvendig underveis i oppgaven. Tirsdager reiser gruppen til Brumunddal og jobber hos oppdragsgiver. Der vil det være et ukentlig møte med oppdragsgiver, samt mulighet til å kommunisere med oppdragsgiver i løpet av dagen. Onsdager starter med møte med veileder, etterfulgt av vanlig arbeidsdag. Torsdag og fredag er vanlige arbeidsdager hvor gruppen møter på avtalt sted og jobber sammen. En vanlig arbeidsdag varer fra 09:00 til 16:00. Hver dag starter med et 15 minutters Daily Scrum møte hvor alle gruppemedlemmer blir enige om hva som er gjort og hva som gjenstår i sprinten.

5 - Organisering av kvalitetssikring

5.1 - Dokumentasjon, standardbruk og kildekode

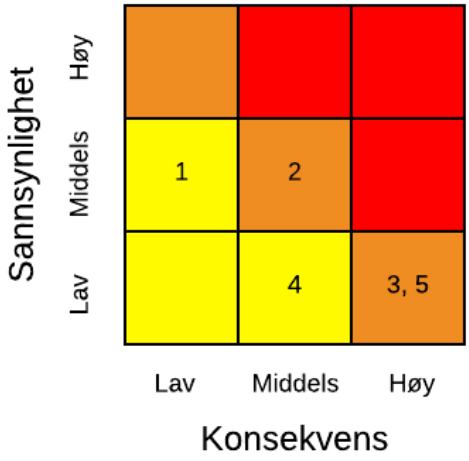
Google Docs brukes hovedsakelig som dokumentasjonsverktøy gjennom prosjektet, men prosjektet kan flyttes til LaTeX underveis skulle det være nødvendig. For å henvise til kilder benyttes Vancouver. All kildekode skal kommenteres slik at det lett kan forstås av andre gruppemedlemmer og oppdragsgiver.

5.2 - Konfigurasjonsstyring

For versjonskontrollering skal gruppen benytte seg av en pakkeløsning fra Atlassian som gir dem tilgang til både Bitbucket og Jira. Gruppemedlemmene skal ha egne grener for oppgavene de jobber med. Når en oppgave er ferdig skal den bli vurdert av de andre gruppemedlemmene før den føres over til hovedgrenen.

Gruppen har sammen med arbeidsgiver valgt å bruke Oracle Jet. Dette er et rammeverk som bruker html, css og javascript til å lage nettsider og hybrid-mobilapplikasjoner. Oracle Jet bruker tjenesten Cordova for å kjøre applikasjoner som mobilapplikasjoner. Alle delene av applikasjonen skal designes slik at de kan kompileres gjennom Oracle Jet sitt kommandolinjeverktøy, utenom dette står gruppemedlemmene fritt til å velge utviklingsmiljø individuelt.

5.3 - Risikoanalyse



Figur 1: Diagram over alvorlighetsgrad for risikoer

	Beskrivelse	Sannsynlighet	Konsekvens	Klassifisering	Tiltak
1	Sykdom	Middels	Lav	Lite alvorlig	God planlegging og kommunikasjon
2	Tidsplan blir ikke fulgt	Middels	Middels	Alvorlig	Ekstra arbeid
3	Gruppemedlem slutter	Lav	Høy	Alvorlig	Justering av oppgaveomfang
4	Tap av arbeid	Lav	Middels	Lite alvorlig	Gode rutiner
5	Svikt i maskinvare	Lav	Høy	Alvorlig	Gode rutiner

Tabell 1: Risikoanalyse

Sykdom er noe en ikke kan forutse, derfor er det viktig at gruppen har god kommunikasjon slik at gruppen skaffer seg god oversikt over hvilke oppgaver som må prioriteres om et medlem ikke kan jobbe. Med god planlegging kan medlemmene av gruppen støtte hverandre og sørge for at konsekvensene blir så små som mulig.

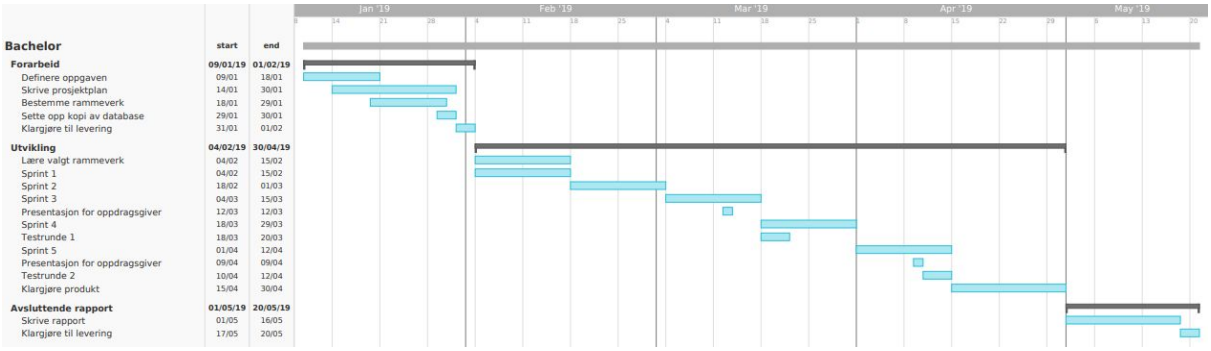
Ettersom gruppen består av studenter med lite erfaring innenfor store utviklingsprosjekter kan det være vanskelig å beregne hvor lang tid det vil ta å løse en enkelt oppgave. Dersom sprintmålene ikke blir nådd kan det ha store konsekvenser senere i utviklingen og det er derfor viktig at gruppen tar kontroll over situasjonen med en gang den ligger bak skjema.

Alle medlemmene på gruppen er svært motiverte for å jobbe med bacheloroppgaven og det svært usannsynlig at noen skulle velge å avslutte eller komme i en posisjon hvor en blir kastet ut. Dersom det allikevel skulle oppstå en situasjon hvor gruppen mister et medlem vil det være mulig å justere ned omfanget av oppgaven slik at prosjektet fortsatt kan fullføres.

Tap av arbeid og svikt i maskinvare er andre typer risikoer som ikke kan forutsees, og disse kan også være vanskelige å fikse opp i om de inntreffer. På grunn av dette er det viktig at gruppen skaffer seg gode rutiner innenfor datalagring og databehandling, slik at dersom noe skjer er det begrenset hvor stor skaden blir.

6 - Plan for gjennomføring

Prosjektet er satt opp til å vare fra 8. januar til 20. mai. Første periode kalles “Forarbeid”, og varer omtrent en måned: fra prosjektstart til innlevering av prosjektplan. Denne perioden brukes primært til å skrive prosjektplan og klargjøre alt som trengs før gruppen kan begynne med utvikling av oppgaven. Andre periode kalles “Utvikling”, og beskriver selve utviklingsprosessen. Denne prosessen er delt opp i Scrum sprinter, som hver varer to uker. I tillegg til sprintene er det også lagt opp til brukertester og presentasjoner for oppdragsgiver i løpet av denne perioden. Utviklingsperioden varer ut april, hvor gruppen går inn i tredje periode, kalt “Avsluttende rapport”. Her er det satt opp til at gruppen skal bruke helt frem til prosjektslutt, 20. mai, til å skrive på rapporten. Rapporten skal også skrives på under andre periode, men da ved siden av selve utviklingen.

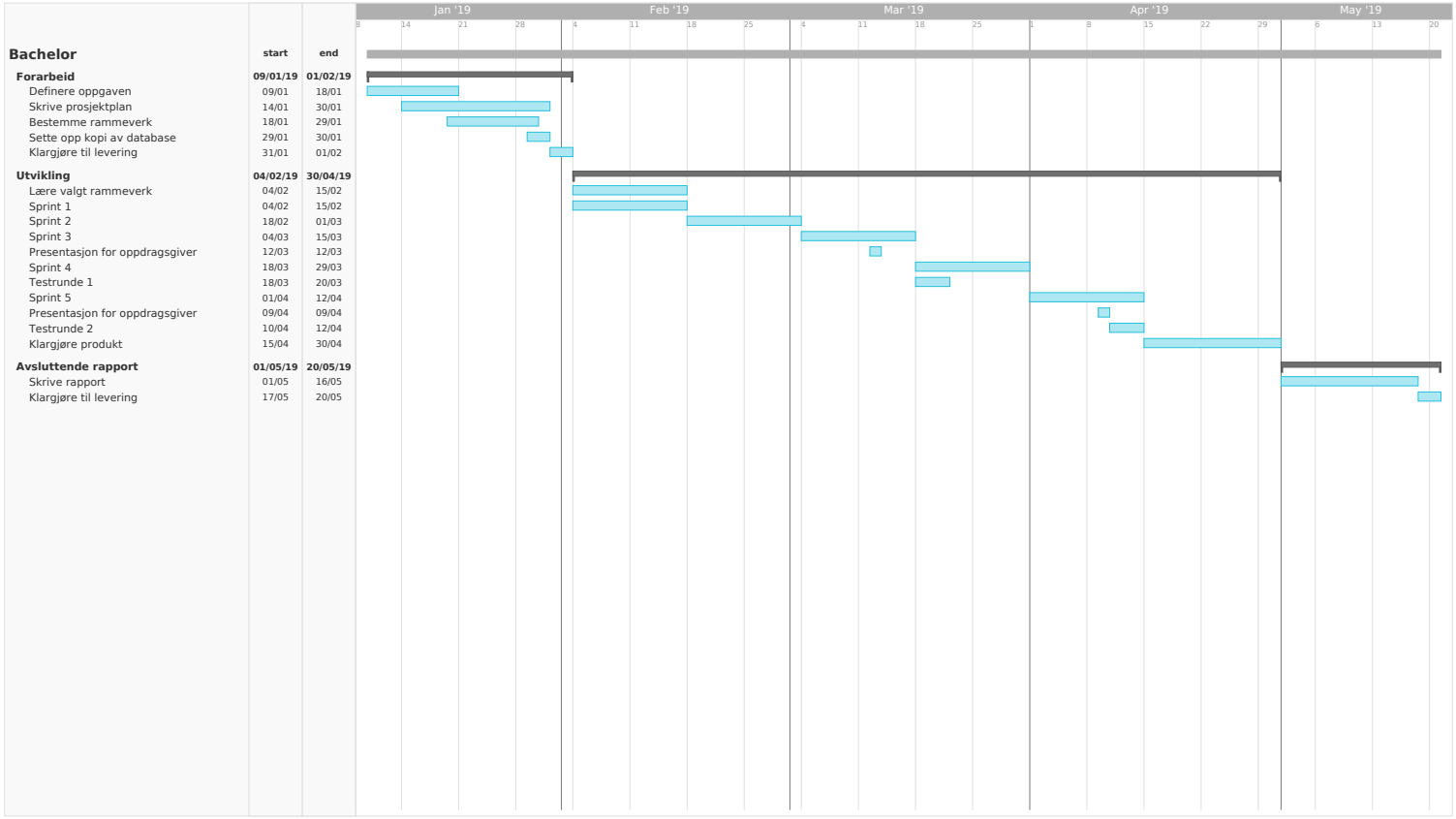


Figur 2: Gantt-diagram

Figur 2 viser et Gantt-diagram som beskriver arbeidsplanen gjennom hele prosjektet. Diagrammet tar høyde for helger, men ikke andre ferie- og helligdager. Sprint 5 vil dermed bli svært annerledes i praksis, da påskeferien varer fra starten av sprinten til omtrent halvveis gjennom. Gantt-diagrammet vil ellers, med unntak av 1. og 17. mai, representere hvilke dager gruppen vil jobbe.

B.1 Gantt-diagram

teamgantt
Created with Free Edition



Bachelor	start	end
Forarbeid	**09/01/19**	**01/02/19**
Definere oppgaven	09/01	18/01
Skrive prosjektplan	14/01	30/01
Bestemme rammeverk	18/01	29/01
Sette opp kopi av database	29/01	30/01
Klargjøre til levering	31/01	01/02
Utvikling	**04/02/19**	**30/04/19**
Lære valgt rammeverk	04/02	15/02
Sprint 1	04/02	15/02
Sprint 2	18/02	01/03
Sprint 3	04/03	15/03
Presentasjon for oppdragsgiver	12/03	12/03
Sprint 4	18/03	29/03
Testrunde 1	18/03	20/03
Sprint 5	01/04	12/04
Presentasjon for oppdragsgiver	09/04	09/04
Testrunde 2	10/04	12/04
Klargjøre produkt	15/04	30/04
Avsluttende rapport	**01/05/19**	**20/05/19**
Skrive rapport	01/05	16/05
Klargjøre til levering	17/05	20/05

C Møtelogger

Her kommer møtenotater fra de større/viktigste møtene med EVRY og veileder.

C.1 Møter med EVRY

26.02.2019

Hadde møte med Merete og Guro fra Evry for å vise appen og få tilbakemelding. Dette var noen av kommentarene og tipsene vi fikk:

- La det stå ”Brukerid (for eksempel telefonnummer)” som fritekst (i placeholder), og ”Brukerid” som overskrift i loggg inn
- Smart å kalle det utilgjengelighet istedenfor tilgjengelighet
- Husk fargekontraster
- Skriv ting som at vi har animasjon på id-kort med en gang i rapporten
- Kall arrangementer tjenester siden arrangementer er en type tjeneste
- Kalender: Arrangementer har fargestatus ut fra null, noen (gul?) eller fulltegna (grønn)
- Ha med flere roller kan vare lurt om vi har tid
- Ha med personvern og sikkerhet i rapporten
- Ta oppgave 1 og heller utvider den enn å gå videre oppgave 2 i forhold til de tre oppgavene vi snakket om i starten av prosjektet
- Skriv ting som møtenotater under daily scrum + andre møtenotater direkte i rapporten

05.03.2019

Hadde sprint review for sprint 2 hvor vi ble enige om at vi hadde blitt bedre til å planlegge sprinten etter sprint 1. Vi ble ferdige med alt som skulle gjøres i sprinten og fikk tid til å skrive på rapporten, slik vi hadde planlagt.

Videre startet vi med sprint 3. Vi hadde et sprint planning møte med oppdragsgiver, og foreslo disse punktene:

- Utvikle ’kalender’ view
- Utvikle ’tjenester’ view
- Forberede tester til test-runde 1
- Implementere varsler
- Dedikere tid til rapporten

Oppdragsgiver foreslo i tillegg disse punktene:

- Bug fiksing
- Bilde-endring
- Farge-endring på knapper

Vi valgte å legge alle punktene inn i sprint 3, med unntak av "Implementere varlser", ettersom vi tenkte at dette kunne ta en del tid og ikke var nødvendig for den første brukertesten i sprint 4.

C.2 Veiledningsmøter

16.01
Svar:

- Avtale flere møter med oppdragsgiver
- Se på liknende applikasjoner
- Se på tidligere liknende bachelorer
- Møte forberedt til møte med Evry, og har tanker om hvordan vi skal løse oppgaven før vi kommer
- Spørre om verktøy vi skal bruke

23.01
Spørsmål:

1. Se på Gantt skjema
2. Spørre om punktene fagområde, konfigurasjonsstyring og hovedinndeling av prosjekt
3. Spørre om vi skal inkludere use cases og kravspec.
4. Spør om uthevet i oppgavebeskrivelse
5. Spørre om innleveringsfrist til planleggingsdokument

Svar:

1. Mer detaljer
2. Beskrive hva FRIDA gjør, avgrensning beskriver hva FRIDA ikke gjør ish. Litt det samme som 5.1 bare med GIT o.l. Om vår bruk av Scrum
3. Nei
4. Ja
5. Se blackboard
6. Skrive om hvordan/om vi tenker å gjennomføre testing av applikasjonen (med Evry / medstudenter?)

30.01

- Bare en skal levere
- Skal ha evry med på både sprint- review og planning

07.02 Spørsmål:

1. Tips i forhold til rapport, hvordan vi bør angripe det?
2. Spørre om om han vet noe mer om Jira

Svar:

1. Forklar valg av rammeverk
2. Dokumentere all valg, hvorfor det ene passet bedre enn det andre
3. Være oppdatert på og ha kunnskaper innen hva de andre på gruppa jobber med
4. Ta bilder av utviklingsverktøy underveis (Jira, Bitbucket, etc)
5. Dokumenter hvordan vi lå an i slutten av hver sprint
6. Hva er målene med den nye appen? (flere brukere, flere kunder, bedre tjeneste?
7. Skrive mer om hvem som skal bruke tjenesten? Typ bedrifter og organisasjoner
8. Forklare mer om roller i gruppa

9. Skrive om testing
10. Mer i risikoanalyse

20.02 Spørsmål:

1. Snakke om hvordan vi ligger an (utvikling)
2. Nevne at vi har byttet til Trello

Svar:

1. Sprint review møter
2. Forklare implementasjonen av passord
3. Dokumentere hvordan vi jobber (alt)
4. Vise helheten av oppgaven i rapporten

20.03 Spørsmål:

1. Bytte av databaseserver samme dag som testing skulle skje, ingen testing
2. Testing utsatt til tirsdag 26. mars
3. Hvordan skal vi takle en slik ufotsigbar utsettelse?

Svar:

1. Dokumentere det vi har gjort til nå, teste teknologi til fremtidige stadier
2. Kan man gjøre det mindre sårbart?

27.03 Spørsmål:

1. Snakke om testen
2. Vise frem appen

Svar:

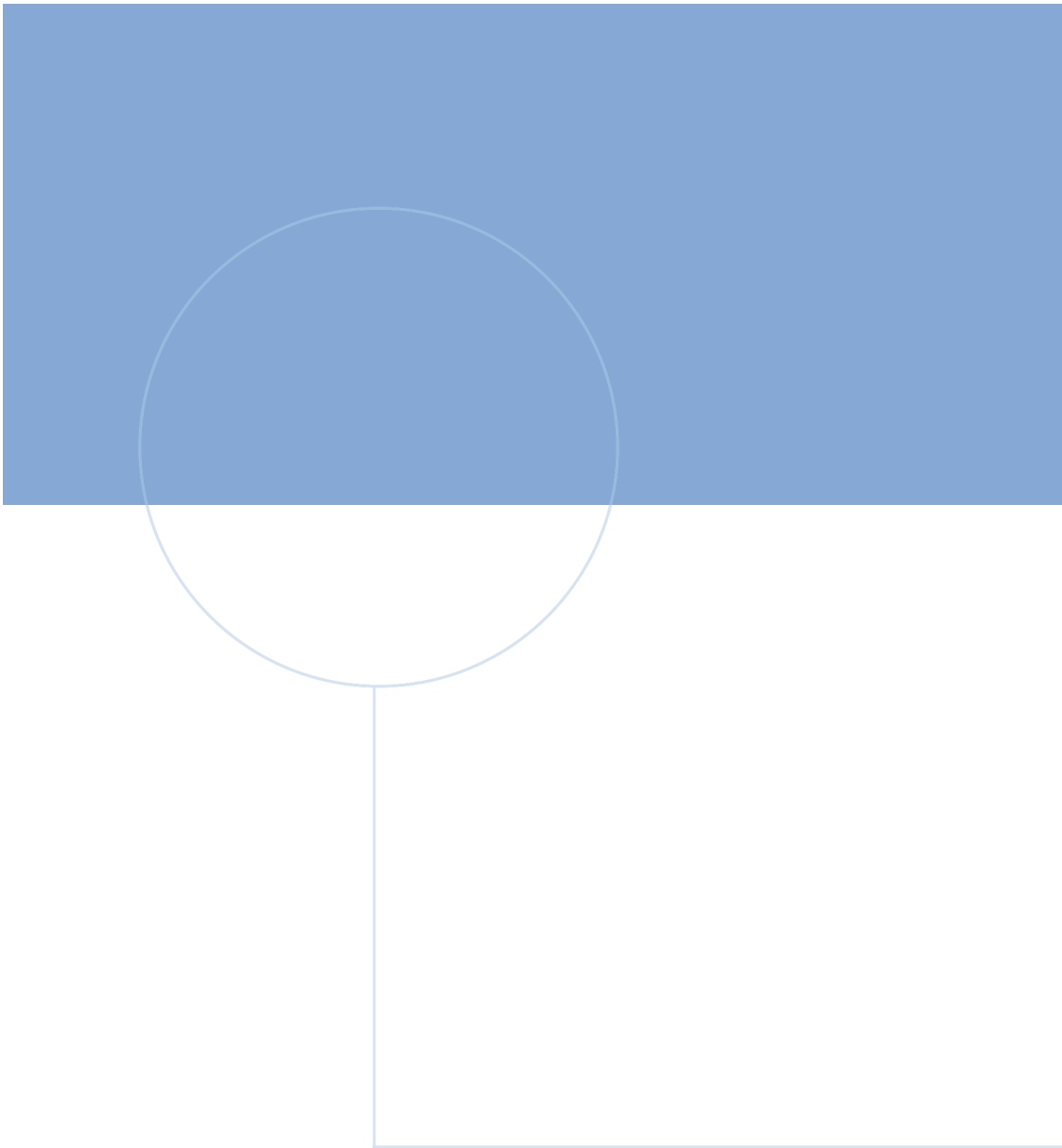
1. Justeringer til legg til interesser": I stedet for at en interesse forsvinner når du legger den til, kan en markere flere interesser og så lagres de når brukeren går tilbake

24.04 Spørsmål:

1. Når Tom trenger utkast til rapport
2. Andre eventuelle datoer som er viktig å huske
3. Spørre om presentasjonsdato

Svar:

1. 10. mai, siste dag Tom er på campus (Møte kl. 09:00)
2. Levere i løpet av 7. mai
3. Sende utkast på epost
4. Repport: Husk å referere til andre deler av rapporten / vedlegg
5. Presentasjondato kommer senere (Evt hør med Tonje)
6. Vi må invitere EVRY
7. Bruk kode aktivt i rapporten



NTNU

Kunnskap for en bedre verden