Bacheloroppgave

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

Johanne Bognøy
Cecilie Urdahl Fossum
Amund Johansen

Spillerstatus

Bacheloroppgave i Bachelor i ingeniørfag - data
Veileder: Tom Røise
Mai 2019

NTNU
Kunnskap for en bedre verden

Johanne Bognøy
Cecilie Urdahl Fossum
Amund Johansen

Spillerstatus

Bacheloroppgave i Bachelor i ingeniørfag - data
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

Sammendrag av Bacheloroppgaven

Tittel:	**Spillerstatus**
Dato:	2019
Deltakere:	Johanne Bognøy Cecilie Urdahl Fossum Amund Johansen
Veiledere:	Tom Røise
Oppdragsgiver:	Gjøvik Idrettsmedisinske Team (GIT)
Kontaktperson:	Terje Løkken, terje@gjovikidrettsmedisin.no, 61135000
Nøkkelord:	Programmering, PWA, Gjøvik Idrettsmedisinske Team
Antall sider:	87
Antall vedlegg:	6
Tilgjengelighet:	Åpen

Sammendrag:

I idretten er en del av jobben til trenere å holde oversikt over spillerne sine og til enhver tid vite hvem som er skadet. Idrettsmedisinere må holde oversikt over klientene sine og koordinere med kolleger og klienter. Med så mange ulike kanaler for kommunikasjon som det er i dag er dette både tungvint og gjør det vanskeligere å opprettholde personvern. Gjøvik Idrettsmedisinske Team ønsket et system for å samle all kommunikasjon for trenere, spillere og helsepersonell på en sikker kanal. Systemet skulle holde oversikt over spillere for trenere, skader for utøvere og informasjon om skader for helsepersonell. Med dette prosjektet har vi levert en prototype som tilnærmer seg dette på en moderne måte og som demonstrerer grunnleggende personvern. Prototypen kan kjøres som et pilotprosjekt og videreutvikles.

Summary of Graduate Project

Title:	**Spillerstatus**
Date:	2019
Authors:	Johanne Bognøy Cecilie Urdahl Fossum Amund Johansen
Supervisor:	Tom Røise
Employer:	Gjøvik Idrettsmedisinske Team (GIT)
Contact Person:	Terje Løkken, terje@gjovikidrettsmedisin.no, 61135000
Keywords:	Programming, Progressive web application, Gjøvik Idrettsmedisinske Team
Pages:	87
Attachments:	6
Availability:	Open

Abstract: In sports, part of coaches' job is to maintain an overview of their players and at all times know who are injured. Sports mediciners need to keep an overview of their clients and coordinate with colleagues and clients. With as many different channels of communication as there are today is this not only cumbersome but it makes protecting privacy more difficult. Gjøvik Idrettsmedisinske Team wanted a system to gather all communication for coaches, athletes and health professionals in a secure channel. The system should give coaches and overview of their athletes, athletes an overview of their injuries and health professionals an overview of the information about injuries. With this project we have delivered a prototype that approaches this in a modern way and that demonstrates basic privacy. The prototype is ready for a pilot project and can be further developed.

Forord

Denne bacheloroppgaven er skrevet av Johanne Bognøy, Cecilie Urdahl Fossum og Amund Johansen ved instituttene for informasjonssikkerhet og kommunikasjonsteknologi (IS) og datateknologi og informatikk (IDI) ved NTNU i Gjøvik. Oppgaven er skrevet for oppdragsgiver Terje Løkken ved Gjøvik Idrettsmedisinske Team. Vi vil takke Tom Røise for god veiledning av bacheloroppgaven denne våren, og hverandre for et godt samarbeid og mange gode diskusjoner. Takk til samarbeidspartner Dag Solhaug ved ETC som også har gitt gode innspill. Til slutt vil vi takke Simon McCallum og Ivar Farup som har laget Bachelor Thesis LaTeX template for NTNU i Gjøvik.

Innhold

Figurer

Tabeller

Ordliste

action (Redux) er en beskrivelse av hvilke endring som skal gjøres på gitt data. 56, 57

action creator (Redux) er en metode som returnerer et objekt med en action og gitt data. 57

asynkron ikke-blokkerende operasjon. En form for I/O-prosessering som tillater at en annen prosess fortsetter før en dataoverføring er avsluttet. 33

attributt Attributter ved en HTML-tag påvirker hvordan HTML-elementet skal oppføre seg. Webkomponenter kan ha egendefinerte attributter. 33, 70, 72

bounce rate Avvisningsfrekvens er en internett-markedsføringsterm som brukes i webtrafikkanalyse. Den representerer prosentandelen av besøkende som går inn på nettstedet og deretter forlater ("sprett") i stedet for å fortsette å se andre sider på samme nettsted. 31

component er en klasse med variabler og metoder som representerer et element i en webapplikasjon. Se webkomponent. 56

container er en isolert prosess som innkapsler kode og avhengigheter for å utgjøre en enhet av programvare. Containere deler vertens OS-kernel og er derfor lettveiende og flyttbare, noe som skiller dem fra virtuelle maskiner (VMer). Containere er en abstraksjon på applikasjonslaget. 40

dispatch (Redux) er en utsendelse av en action til store. 57

DOM Document Object Model. En API for å aksessere elementer i et HTML-dokument. 33, 34

DPIA Data Protection Impact Assessment, på norsk vurdering av personvernkonsekvenser. Pålagt å gjennomføre når ”behandlingen [av personopplysninger] sannsynligvis vil medføre en høy risiko for fysiske personers rettigheter og friheter”. [1]. 11

ECMAScript er en programmeringsspråk-spesifikasjon. JavaScript følger denne standarden. 34

FCM Firebase Cloud Messaging. Tjeneste fra Google for push-varslinger. 57

geolokasjon er identifikasjon eller estimering av den geografiske plasseringen av et objekt, for eksempel en radarkilde, mobiltelefon eller Internett-tilkoblet dataterminal. 29

HTML-tag definerer et element i HTML som en slags kommando. Kan ses på som en node i dokumenttreet. x, 35, 51

hybrid-applikasjon er en web-applikasjon vist i et web-view gjennom et rammeverk. Det er en krysning mellom web-applikasjoner og native-applikasjoner, og kan lastes ned på telefonen. 29

i18n Internasjonalisering og lokalisering omfatter prosessen for å forberede programvare og nettsteder for et internasjonalt marked, og tilpasse den til lokale forhold i ett land eller språk. Prosessen omfatter tilpassing av tekster, sorteringsfølger, datoformat og så videre. I forkortelsen i18n står 18 for antall bokstaver mellom første i og siste n i 'internationalization'. 52

image (Docker) er en utgave av en applikasjon i en fil som kan kjøres i en container. 40

lazy loading er et prinsipp i webutvikling som betyr at man laster inn ressurser etter hvert som de trengs eller blir forespurt. 32

manifest Manifestet til en webapplikasjon gir informasjon om applikasjonen (for eksempel navn, forfatter, ikon og beskrivelse) i en JSON tekstfil. Manifestet informerer om detaljer for nettsteder installert på hjemmeskjermen til en enhet, noe som gir brukerne raskere tilgang og en rikere opplevelse. 32

markup er en deklarativ tekst som bruker tagger til å definere elementer. 37

native applikasjon er en nedlastbar applikasjon som har tilgang til enhetens maskinvare. 2, 29

NPM Node.js Package Manager, et verktøy for å bygge Node-applikasjoner samt administrere Node-pakker. 44, 53, 70

polyfill er kode som implementerer funksjonalitet på nettlesere som ikke støtter funksjonaliteten av seg selv. 39

PRPL er et konsept innen webutvikling som står for Push, Render, Pre-cache og Lazy-load. 32

PWA Progressive Web Application, på norsk progressiv nettapplikasjon. Bygget på moderne prinsipper for webutvikling. 29, 32

rating of perceived exertion er ofte brukt i forbindelse med idrett og spesielt under testing på trening, for kvantitativ måling av oppfattet anstrengelse av fysisk aktivitet. Innen medisin brukes dette for å dokumentere pasientenes anstrengelse under en test, og trenere bruker dette til å vurdere intensiteten av trening og konkurranse. 7

reducer (Redux) er en metode som gjør endringer på tilsendt kopi av en state. 56, 57

reverse proxy er en mellomtjener som sender klientforespørsler til riktig applikasjon på serveren og returnerer responsen deres, slik at det ser ut som om responsen kom fra mellomtjeneren. 41, 61

service worker er et skript som nettleseren kjører i bakgrunnen, noe som åpner for funksjonalitet som ikke krever brukerhandling. Pushvarsler benytter service workers. 29, 32, 41, 48, 49, 67

splash screen er et skjermbilde som kommer opp når f.eks. et program lastes inn. Består som regel av enkel grafikk/bilde og versjonsnummer. 29

state (Redux) er hele applikasjonens sannhet. Alle data lagres kun et sted; i state. xii, 32, 56

store (Redux) er lagringsenheten for state. x, 32, 56

template er en ferdiglaget HTML-mal som angir strukturen til et HTML-dokument. 33, 53

triage (fransk for sortering) er en metode for å bestemme prioritering i behandling av pasienter basert på hvor alvorlig deres medisinske tilstand er. Dette skal gjøre pasientbehandlingen mer effektiv når ressursene er utilstrekkelige til at alle kan bli behandlet med en gang. [2]. 7

Vanilla Javascript er en betegnelse på standard JavaScript uten bruk av rammeverk. 2, 33

webkomponent er en egendefinert HTML-tag som kan gjenbrukes, og som innkapsler sin egen kode. x, 33, 34, 39, 49

Kodeliste

1 Introduksjon

1.1 Problemområde

I digitaliseringens tid har vi fått mange plattformer å kommunisere på. Digital kommunikasjon er blant annet ment til å effektivisere folks arbeid, men med så mange konkurrerende kanaler er ikke dette alltid tilfelle. I idretten, for eksempel, er en del av jobben til trenere å holde oversikt over spillerne sine og til enhver tid vite hvem som er skadet. Når kommunikasjonen mellom trener og spillere er spredt på ulike kanaler som muntlig samtale, SMS, telefon og ulike sosiale medier, der ikke alle er loggført, er det tungvint å holde en slik oversikt.

Slik er det også for helsepersonell. I idrettsmedisin jobber ofte helsepersonell tverrfaglig, og må derfor koordinere med både kolleger og klienter. Helsepersonell har et stort ansvar i å behandle skadetilstander hos klientene sine. Men som helsefagarbeidere plikter de samtidig å opprettholde personvern, og derfor er det viktig at kommunikasjonen deres skjer på en sikker kanal.

Gjøvik Idrettsmedisinske Team (GIT) er en tverrfaglig faggruppe innenfor kiropraktikk, medisin og fysioterapi. De jobber tett sammen for å finne gode løsninger for den enkelte idrettsutøver for å stille riktig diagnose og iverksette tiltak for at utøveren skal komme raskt tilbake til idretten. I deres daglige arbeid opplever de at det er dårlig beskyttelse av personopplysninger. De observerer at sensitive opplysninger ofte sendes over epost og SMS, hvor identiteten til idrettsutøvere kun skjules ved f.eks å bruke deres initialer. Personopplysninger spres altså på forskjellige usikre kanaler. Dette er både dårlig personvern og gir dårlig oversikt over spillernes status for helsepersonell og trenere.

GIT uttrykker behov for en kommunikasjonsløsning mellom medisinsk helsepersonell, trenere og utøvere. Dette er for å kunne gi helsepersonell og trenere en samlet oversikt over spillernes status på en sikker måte, og tilby en sikker kanal for utøvere å rapportere skader på. For helsepersonell skal løsningen ikke bare tilby en oversikt, men være en administrasjonsplattform. Videre kan tidlig informasjon om smertetilstander for helsepersonell danne grunnlag for skadeforebyggende tiltak og kunne redusere skadetid. Trenere vil kunne få en oppdatert oversikt over utøvernes status i forhold til skade og om de er konkurranseklare, og dette vil også kunne hjelpe dem å legge opp til tilrettelagte treninger for utøverne ved behov.

1.2 Avgrensning

Kommunikasjonen i denne løsningen skal ikke være en konkurrent til andre kommunikasjonsplattformer slik som sosiale medier, men er ment som korrespondanse knyttet til hver enkelt skade en utøver har, mellom utøveren og helsepersonell.

Vår samarbeidspartner i ETC rådet oss til å lage en webapplikasjon heller enn en native applikasjon, for dette er den mest tilgjengelige løsningen. Systemet skal derfor bestå av en webapplikasjon som skal være tilgjengelig på PC, nettbrett og mobiltelefon, og vi skal ikke lage en native løsning for mobil slik som en Android- eller iOS-app. Vi skal fokusere på å lære oss gode utviklingsmetoder med Vanilla Javascript og uten rammeverk så langt det lar seg gjøre, og etterstrebe utvikling av en en webapplikasjonen som er intuitiv og lett tilgjengelig for brukerne.

Applikasjonen vi lager skal være en prototype som skal kunne danne grunnlag for et pilotprosjekt. Det er viktig at behandlingen av personopplysninger i appen kvalitetssikres før appen settes ut i produksjon. Sikkerheten vi bygger inn i appen skal være god nok for et pilotprosjekt, men det er utenfor dette prosjektets omfang å gjøre den produksjonsklar med tanke på graden av sensitive opplysninger den skal beskytte.

1.3 Oppgavedefinisjon

Vår oppgave er å levere et webgrensesnitt for bruk i det daglige samarbeidet mellom helseforetak som GIT, idrettsutøvere og trenere. Systemet skal tilby en måte å holde oversikt over skader på for helsepersonell, trenere og utøvere samt en sikker kanal for kommunikasjon.

Helsepersonell, både enkeltpersoner og foretak, skal kunne bruke appen med de idrettslag de har avtale med. Når en utøver registrerer en ny skade skal helsepersonell bli varslet om dette. De kan da avtale time for utredning, og helsepersonell setter spillerstatusen til utøveren.

Trenere og helsepersonell skal få en til enhver tid oppdatert oversikt over spillerstatus til utøverne på alle idrettslag de er knyttet til. Helsepersonell skal kunne gå inn på hver utøver og se personinformasjon samt administrere skader og behandlingsplan for vedkommende.

Utøvere skal kunne rapportere inn nye skader og legge inn grunnleggende endringer på eksisterende skader, slik som egen beskrivelse. De skal få en oversikt over skadehistorikk og se sin egen spillerstatus.

Helsepersonell og utøvere skal ha en meldingskanal per skade, i tillegg til en generell

kanal. Utøvere har altså en innboks per skade hvor alt helsepersonell tilknyttet laget til utøveren har tilgang. Det skal også være en meldingskanal mellom utøver og alle tilknyttede helsepersonell og trenere. Denne kanalen er den eneste for trenere.

Vi definerer følgende roller i systemet: Helsepersonell, Idrettsklubb, Utøver og Trener. Helsepersonell og Idrettsklubb klassifiseres som administratorroller, mens Utøver og Trener er vanlige brukerroller.

Brukergrensesnittet skal være forskjellig for hver rolle, og applikasjonen skal ha følgende overordnede funksjonalitet for de forskjellige rollene:

Rolle	**Funksjon**
Alle	Registrere ny bruker
	Sikker innlogging
Helsepersonell	Registrere helseforetak i systemet
	Oversikt over utøvere
	Registrere en skade for en utøver
	Oppdatere en skade hos en utøver
	Endre en utøvers spillerstatus
	Administrere behandlingsplanen for en utøver
	Meldingsutveksling med utøver og med trener
Utøver	Registrere skade
	Oversikt over nåværende og tidligere skader
	Endre grunnleggende informasjon om en skade
	Meldingsutveksling med helsepersonell og med trener
Trener	Lagoversikt med spillerstatus
	Meldingsutveksling med helsepersonell og utøver
Idrettsklubb	Registrere idrettsklubb og lag i systemet
System	Loggføring

Tabell 1: Funksjoner per rolle i systemet

1.4 Formål

Systemet skal:

- Gjøre det enklere for helsepersonell og trenere å holde oversikt over spillerstatus på utøverne i et lag
- Effektivisere skaderapportering og -oppfølging
- Fange opp smertetilstander raskt og dermed forebygge skader mer effektivt

- Kunne redusere skadetid og skaderate
- Gjøre det enklere for helsepersonell å følge personvernloven
- Forbedre rutiner og sikkerhet omkring taushetsbelagte opplysninger i helseforetak

1.5 Målgruppe

Systemet har tre målgrupper: utøvere, trenere og helsepersonell.

Utøvere kan være medlemmer av et idrettslag eller selvstendige idrettsutøvere. Vi forutsetter at et idrettslag er underordnet en idrettsklubb.

En trener har ansvar for et idrettslag, eller er personlig trener for en eller flere utøvere.

Med helsepersonell mener vi helsefagarbeidere som er ansatt i enten et foretak eller en idrettsklubb.

1.6 Bakgrunn og kompetanse

Det er ulik kompetanse i gruppen vår. Johanne studerer bachelor i IT-drift og informasjonssikkerhet, og har derfor en bakgrunn i sikkerhet ved tilnærming til utviklingsprosjekter. Cecilie og Amund studerer bachelor i ingeniørfag - data, som har gitt et generelt grunnlag i programvareutvikling. Ingen på gruppen har hatt fokus på frontend i utdanningen, men alle har kjennskap til HTML og CSS fra før av.

Alle har hatt emnet IMT2243 Systemutvikling, som vil være nyttig først og fremst i planleggingsfasen men også gjennom hele den overordnede prosessen. Cecilie har hatt emnet IMT3110 Programvaredesign, som vi også vil få bruk for i planleggingsfasen. Videre har alle vært gjennom IMT2571 Datamodellering og databasesystemer, som er essensielt for databasedesignet. Emnet IMT3281 Applikasjonsutvikling, som Johanne og Cecilie har hatt, vil være nyttig ettersom vi har lært Docker og Docker Compose for oppsett av utviklings- og produksjonsmiljøer. I tillegg har Johanne hatt Docker som pensum i emnet IMT3003 Drift av tjenestearkitekturer.

Følgende områder er nytt for alle på gruppen: JavaScript, Polymer, Node.js og Express. Johanne og Cecilie har emnet IMT2291 WWW-teknologi ved siden av dette prosjektet, der JavaScript og Polymer vil være læringsområder i andre halvdel av semesteret. Vi ser at vi må begynne å lære disse på egenhånd ettersom utviklingsfasen vil starte før vi kommer til disse i IMT2291.

1.7 Rammer

Juridiske rammer

Systemet skal være i tråd med personopplysningsloven [3], og derav leve opp til kravet om innebygd personvern. Sistnevnte må involveres gjennom hele systemutviklingsprosessen.

Datatilsynet klassifiserer helseopplysninger som sensitive [4]. Derfor er vi pålagt å implementere tofaktorautentisering på innlogging som gir adgang til helseopplysninger.

Teknologiske rammer

Webgrensesnittet skal fungere på på de mest populære nettleserne per 31. desember 2018. [5] Det vil si nyere versjoner av Chrome, Safari, Edge, Firefox og Opera. Disse støtter HTML 5 og ECMAScript 5 (Javascript versjon 5) [6].

Vi velger å ikke støtte Internet Explorer. Dette fordi vi tror det ville blitt for mye merarbeid i å optimere webapplikasjonen for denne nettleseren, og for å bidra til at den erstattes av Edge.

Økonomiske og tidsmessige rammer

Programvaren skal være ferdig i følge Gantt-diagrammet i prosjektplanen, B. Vi har blitt enige med oppdragsgiver at vi i utviklingen skal prioritere funksjonaliteter.

Utgifter til pilotprosjektet etter dette prosjektets slutt dekkes av GIT.

1.8 Om rapporten

Prosjektplanen er vedlagt, B. Denne er uendret siden 1. februar 2019, og mye av innholdet er derfor utdatert; alle oppdateringer av planen, valg og forutsetninger har blitt gjort i denne rapporten.

Når vi bruker ordet ”systemet” mener vi hele løsningen vår. ”Systemet” er et overordnet ord som omfatter både klient, server og database. Synonymt med dette er ”løsningen”, ”applikasjonen” og ”appen”.

1.8.1 Rapportens struktur

Rapporten inneholder syv kapitler. Med unntak av innledningen inneholder disse:

- Forarbeid vi gjorde før vi begynte å utvikle applikasjonen
- Kravspesifikasjon på bakgrunn av forarbeid
- Valg av teknologier for systemet
- Utviklingsprosessen og hvordan vi gjennomførte den i forhold til planen

- Design og implementering av løsningen
- Brukergrensesnitt for ferdig løsning
- Konklusjon med resultat, videre arbeid og drøfting

2 Forarbeid

2.1 Kartlegging av lignende systemer

Det finnes i dag noen ulike systemer som brukes av idrettsutøvere, trenere og idrettsklubber i Norge. De varierer i omfang av funksjonalitet, men alle har til felles at de tilbyr ulike verktøy for bruk i det daglige arbeidet innen idrett. Vi har valgt å se litt nærmere på to av disse systemene og hva de tilbyr sammenlignet med systemet vi skal utvikle. Dette utvalget er basert på oppdragsgivers kontaktnettverk, GIT ved Terje Løkken, om hva som er i bruk.

Kit Management (Kit)

Kit sitt system er bygget opp av ulike moduler for å gi fleksibilitet til å skreddersy løsninger for idrettslag eller enkeltutøvere. [7] Disse er:

RTP - Return to Play Journalsystem med muligheter for standarisert registrering av skader og sykdommer, sikker kommunikasjon og triage.

Team Generelle notater, oppslagsverk av dokumenter, medikamentregistrering og historikk.

Daily Report Innhente subjektive og objektive data for statistikk og belastningsstyring. Data hentes fra utøvere via mobile løsninger eller ved manuell registrering. Inneholder bl.a. rating of perceived exertion.

UEFA Årlig lisensundersøkelse i skjematisk utforming - hvor data lagres enkelt. Gir lett innsikt og hvor repeterende tester kan hentes frem. Tidligere historikk ligger et tastetrykk unna.

Landslaget i fotball benytter seg av Kit.

XPS Network (XPS)

XPS Network er en modularisert programvare for analyse, planlegging og kommunikasjon innen idrett og trening. [8] XPS tilbyr fire typer pakker: for trenere, for lag og klubber, for akademier, landslag og organisasjoner og for personlige trenere. Disse pakkene inneholder litt ulik sammensetning av forskjellige verktøy. Verktøy som tilbys:

Samle alt arbeidet på et sted Bygg opp og organiser dine øvelser og dokumenter. Ditt materiell finnes alltid tilgjengelig når som helst, hvor som helst.

Kraftig verktøy for treningsplanlegging Planlegg raskt og enkelt fra din øvelsessamling. Del trening og analyse med utøverne direkte via web og mobil.

Direkte tilgang til viktig informasjon Enkel tilgang på trening og konkurransehistorikk fra dine nåværende og tidligere lag og utøvere, klar til fordypning på sekunder.

Øyeblikkelig tilgang til viktig informasjon Enkel tilgang til statistikk på treninger, individuell utvikling, tester og ernæring.

Hold alle oppdaterte Del enkelt informasjon med utøvere og trenere. XPS Network kan brukes av en trener, et trenerteam eller flere tusen sammenkoblede trenere og utøvere.

Følg flere kunder Hjelp dine kunder til å bli mer selvstendige, for deretter å bruke XPS til å veilede når dere ikke er sammen.

Effektiviser Bygg opp og organiser dine øvelser og dokumenter. Ditt materiell finnes alltid tilgjengelig når som helst, hvor som helst.

Perfekt verktøy for treningsplanlegging Planlegg raskt treninger og del via web eller telefon. Tilgjengelig for utøverne når de trenger det.

Profesjonell veiledning Del motiverende rapporter og grafer som viser utøvernes progresjon direkte via web og telefon.

XPS har blitt benyttet noe av aldersbestemte landslag i ulike idretter.

Funn

Da vår oppdragsgiver først oppdaget disse systemene ble vi alle litt bekymret. Men det viste seg at ikke mange bruker disse systemene, sannsynligvis fordi de koster for mye. Likevel er det reelt behov som disse systemene forsøker å møte. Løsningen vi kommer til å lage vil være svært liten og lett sammenlignet med disse, og kanskje det er en bedre tilnærming. Uansett bestemte vi oss for å ta litt inspirasjon fra disse systemene i forhold til funksjonalitet.

Kits Return to Play-løsning gir en ganske god beskrivelse av det vår løsning skal gjøre. Vi tror dette er essensen av behovet til idretten, i alle fall GIT. Forskjellen er at vi ikke ønsker å bygge et journalsystem inn i appen. Ettersom GIT bruker et eget journalsystem, vil vi holde det mulig i fremtiden å integrere vår løsning med et slikt system. Beskrivelsen over nevner også triage, noe vi ikke har nevnt spesifikt i oppgavebeskrivelsen, men som vil kunne muliggjøres indirekte i skaderapportering.

Vi ønsker å få med enkel statistikk som en del av lagoversikten. Dette begrenser vi til antall utøvere i spill og antall ute med skade, som vil gi en til enhver tid oppdatert

skaderate for laget. Vi ser stor nytte i å føre statistikk, blant annet på grunn av disse systemenes vektlegging av dette, men tidsbegrensningen vår gjør at vi må holde det minimalt.

2.2 Spørreundersøkelse

Siden systemet skal resultere i et pilotprosjekt og det er usikkerhet rundt hvilken funksjonalitet som burde være med, har vi valgt å gjennomføre en spørreundersøkelse. Vi forventer at undersøkelsen kan gi svar på noe av hva trenere, utøvere og helsepersonell ønsker seg i et slikt system.

Spørreundersøkelsen vil også kunne gi svar på hvilke verktøy som brukes for å holde oversikt over skader i et idrettslag, og hvordan kommunikasjon rundt dette foregår.

Valg av metode og utforming

Ingen av oss har erfaring med intervjuer, men vi har gjennomført enkle spørreundersøkelser med skjema før på skolen. Vi har også lest litt om spørreundersøkelser i [9] og [10]. Vi ønsker svar fra flere idrettslag for å få et inntrykk av hva de fleste mener, og det er enkelt å nå ut til mange ved å sende et skjema. Derfor går vi for en kvantitativ spørreundersøkelse.

Vi bruker skalaer med fire alternativer for å unngå at de som svarer legger seg nøytralt på alternativet i midten. På denne måten må de som svarer ta en side.

Spørsmålene er formulert for å få deltakerne til å se for seg scenarier ved bruk av vår app, slik at de lettere kan si hvorvidt de ønsker seg en slik app.

Spørreundersøkelsen er anonym, men vi spør om den som svarer har rollen utøver, trener eller helsepersonell fordi spørreskjemaet er tilpasset hver rolle.

Utvalgsmetoder

I samarbeid med oppdragsgiver har vi kommet frem til fire lag i området hvor det er sannsynlig at vi får svar ved hjelp av kontaktnettverk. Av disse lagene er to i eliteserier innen fotball og håndball.

Utsending

Vi utarbeidet spørreundersøkelsen i Google Skjemaer, for så å hente en link som kan deles for å sende ut skjemaet.

Vi kontaktet lagene vi ville ha svar fra og spurte om lov til å sende de et spørreskjema, før vi sendte den. Spørreundersøkelsen ble sendt ut gjennom en kontaktperson i hvert lag. Denne kontaktpersonen tok ansvar for å dele spørreundersøkelsen på en passende måte for hvert lag.

Ved å gå inn på spørreundersøkelsen i Google Skjemaer kunne vi se alle svarene som hadde kommet inn og få statistikk på disse.

Testing

Vi gjennomførte en testrunde av selve spørreskjemaet før vi sendte spørreundersøkelsen ut til hovedgruppen. Testgruppe for utøvere og trenere var Gjøvik Swans ved Vind idrettslag, og testgruppe for helsepersonell var kollegene til oppdragsgiver i GIT. Vi spurte testgruppen om spørsmålene til slutt, og om de synes undersøkelsen var relevant for dem.

Vi fikk 30 svar i testrunden, noe vi var veldig fornøyd med. Etter testrunden endret vi noen av spørsmålene for å være mer tydelig, da vi så at det ble noen misforståelser rundt enkelte spørsmål. Vi sendte så ut skjemaet til hovedgruppen.

Resultat fra spørreundersøkelsen

Etter bare 7 svar i hovedrunden valgte vi å slå disse svarene sammen med de fra testrunden, undersøke disse og så avslutte arbeidet med spørreundersøkelsen.

Svarene vi fikk tydet på at mye av informasjonen rundt idrettsskader utveksles over usikre kanaler, noe vi hadde mistanke om. Utøvere kontakter trenere på mange ulike måter for å melde om og diskutere skader, og det er liten bevissthet rundt informasjonssikkerhet.

Svargruppen var generelt positive til ideen om appen vi la fram, og det kom noen individuelle forslag til funksjoner appen kunne inneholde. Et av disse var oversikt over posisjoner for trenere i idretter som har disse. På grunnlag av dette valgte vi å inkludere posisjon og draktnummer i oversikten over utøvere når det er relevant. Vedkommende ønsket seg en visualisering av spillebanen med posisjoner, men dette vurderte vi som ikke viktig nok.

Alt i alt kunne vi ha vært mer effektive dersom vi hadde droppet spørreundersøkelsen, men vi er likevel glad for at vi gjennomførte den ettersom vi fikk bekreftelse på at det er et behov for systemet vi vil lage.

2.3 Personvern

Datatilsynet setter strenge krav til behandling av personopplysninger. Ettersom løsningen vår skal behandle helseopplysninger, noe som er sensitiv informasjon med høy risiko for enkeltpersoner, er det ifølge Datatilsynet pålagt [1] å gjennomføre en vurdering av personvernkonsekvenser (DPIA) før behandlingen starter.

Vi har undersøkt Datatilsynets veiledere for gjennomføring av en DPIA og for programvareutvikling med innebygd sikkerhet [11], der DPIA er et av stegene. Vi forstår det slik at selv om kravene gjelder for alle, er disse veilederne ment for dedikerte team i større

utviklingsprosjekter. Vi har ikke kapasitet i dette prosjektet til å gjennomføre en komplett DPIA eller følge hele veilederen for innebygd sikkerhet. Derfor må behandlingen i løsningen vår utsettes til det har blitt gjort en DPIA; dette kommer vi tilbake til.

I Datatilsynets veileder for gjennomføring av en DPIA står det om når det er pålagt å gjennomføre en slik risikovurdering. Samtidig nevnes det at det også må gjøres en risikovurdering av personopplysningssikkerhet, og at denne er pålagt for alle behandlinger. Dette er altså en mer grunnleggende vurdering av sikkerhetsrisiko, der verdiene er personopplysninger. I kapittel 4 av Vurdering av personvernkonsekvenser [1] står det at ”Risikovurderingen skal identifisere områder som kan medføre utilsiktet eller uautorisert (ulovlig) tilgang, endring, sletting, tap eller utlevering av personopplysninger”. Til forskjell er verdiene i en DPIA personers rettigheter og friheter.

Prosjektet vårt skal lede til et pilotprosjekt. Dette pilotprosjektet vil finne sted etter at vi har overlevert programvaren til oppdragsgiver, og vil bestå av at en testgruppe prøver ut løsningen vår. Dette kan danne grunnlag for videreutvikling av løsningen. Ettersom et slikt pilotprosjekt vil innebære behandling av personopplysninger, og vi ikke kan gjennomføre en DPIA, må det ikke brukes opplysninger om reelle personer i pilotprosjektet. Det må gjennomføres en ordentlig DPIA ved videreutvikling og senest innen produksjonssetting. Vi velger å kun gjøre en grunnleggende risikovurdering i dette prosjektet.

Risikovurdering av personopplysningssikkerhet

I denne risikovurderingen bruker vi en skala 1-4 på verdi, sannsynlighet og konsekvens. Vi definerer sannsynlighet som sjansen for at en bestemt hendelse vil skje i løpet av et år.

	Verdi	**Sannsynlighet**	**Konsekvens**
1	Liten	Usannsynlig	Liten
2	Middels	Noe sannsynlig	Middels
3	Stor	Sannsynlig	Stor
4	Svært stor	Svært sannsynlig	Katastrofal

Tabell 2: Skala for risikovurdering

Vi går ut fra at følgende personopplysninger lagres i systemet for alle brukere:

- Fullt navn
- Epostadresse
- Telefonnummer
- Fødselsdato
- Fødselsnummer

For utøvere antar vi at følgende opplysninger lagres i tillegg:

- Skader, behandlingsplan, evt. posisjon, evt. draktnummer
- Skader består av: tittel, beskrivelse, registreringstid, starttid og forventet sluttid.
- Behandlingsplaner består av: ansvarlig helsefaglig, starttid, sluttid, instruksjoner og medisiner.

Verdi	**C**	**I**	**A**	**Totalt**
Behandlingsplan	3	4	4	11
Skade(r)	3	4	3	10
Fødselsnummer	3	3	3	9
Telefonnummer	2	2	2	6
Fullt navn	1	2	2	5
Epostadresse	1	2	2	5
Fødselsdato	2	1	1	4
Posisjon	1	1	1	3
Draktnummer	1	1	1	3

Tabell 3: Verdivurdering

Verdi Personopplysningen som verdien representerer
C Konfidensialitetsnivået på verdien
I Integritetsnivået på verdien
A Tilgjengelighetsnivået på verdien
Totalt Verdiens samlede vurdering (summen av C, I og A)

Trusselaktør	**Motivasjon**	**Intensjon**	**Angrepsvektor**	**Sannsynlighet**
Kriminell aktør	Sabotasje eller økonomisk vinning	Identitetstyveri	Social engineering og/eller innbrudd i systemet for å uthente opplysninger	2
Helsepersonell	Sabotasje eller N/A ved uhell	Lekke sensitiv informasjon om idrettsutøver(e)	Misbruk av tilgang til sensitive opplysninger: deling av disse med uvedkommende	2
Konkurrerende idrettsklubb, annen kriminell aktør eller helsepersonell	Sabotasje	Uriktig endring på viktige opplysninger	Innbrudd i systemet eller misbruk av tilgang	1
Konkurrende idrettsklubb eller annen kriminell aktør	Sabotasje eller annet destruktivt motiv	Tilgangsnekt	DDoS-/skadevareangrep mot systemet	1

Tabell 4: Trusselvurdering

Trusselaktør Personen(e) som utgjør trusselen
Motivasjon Trusselaktørens bakgrunn for å starte et angrep
Intensjon Trusselaktørens ønskede utfall av angrepet
Angrepsvektor Metode trusselaktøren benytter seg av
Sannsynlighet Hvor sannsynlig det er at trusselen realiseres i løpet av et år

Beskrivelse	Sårbar verdi	Angrepsvektor	Angrepskrav	Eksponering
Menneskelige feil	Personopplysninger, integritet	Angriper utnytter registrertes rutiner, vaner, naivitet eller ønske om å hjelpe for å få tak i konfidensiell informasjon	Manglende bevissthet hos registrerte	2
Sikkerhetsfeil ved systemet	Personopplysninger og potensielt konfidensialitet eller integritet	Angriper bryter seg inn i systemet	Erfaring og kunnskap innen hacking	2
Menneskelige feil	Personopplysninger, konfidensialitet	Helsepersonell lekker ved et uhell utøvers personlige informasjon	Manglende bevissthet hos helsepersonell	2
Dårlige intensjoner hos registrert helsepersonell	Personopplysninger, konfidensialitet	Helsepersonell lekker med vilje utøvers personlige informasjon	Interesse/ønske om sabotasje	1

Tabell 5: Sårbarhetsvurdering

Beskrivelse Beskrivelse av sårbarheten
Sårbar verdi Verdien som er tilgjengelig via sårbarheten
Angrepsvektor Måten sårbarheten utnyttes
Angrepskrav Krav eller forutsetning for å utnytte sårbarheten
Eksponering Sannsynligheten for at sårbarheten utnyttes i løpet av et år

1	
Trussel	Kriminell aktør eller konkurrende idrettsklubb
Sårbarhet	Menneskelige feil
Handling	Angriper benytter social engineering til å få tak i påloggingsinformasjon til en bruker av systemet
Verdi	Personopplysninger og integriteten til systemet
Motivasjon	Økonomisk vinning eller sabotasje
Utfall	Identitetstyveri: angriper kan uautorisert logge seg på i systemet og kan f.eks. sende meldinger som en annen person

Før tiltak: Sannsynlighet: 3 Konsekvens: 3 Risiko: 9
Etter tiltak: Sannsynlighet: 2 Konsekvens: 3 Risiko: 6
Spesifikke tiltak: Tofaktorautentisering

2	
Trussel	Kriminell aktør eller konkurrende idrettsklubb
Sårbarhet	Sikkerhetsfeil ved systemet
Handling	Aktør gjør et DDoS-/skadevareangrep mot systemet
Verdi	Tilgjengeligheten til systemet
Motivasjon	Sabotasje eller annet destruktivt motiv
Utfall	Tap av tilgjengelighet inntil angrepet er avverget, potensielt tap av data i systemet

Før tiltak: Sannsynlighet: 2 Konsekvens: 4 Risiko: 8
Etter tiltak: Sannsynlighet: 1 Konsekvens: 4 Risiko: 4
Spesifikke tiltak: DDoS-beskyttelse

3	
Trussel	Kriminell aktør eller konkurrende idrettsklubb
Sårbarhet	Sikkerhetsfeil ved systemet
Handling	Angriper utnytter sikkerhetsfeil til å bryte seg inn i systemet
Verdi	Personopplysninger
Motivasjon	Økonomisk vinning eller sabotasje
Utfall	Angriper får uautorisert tilgang til personinformasjon; tap av konfidensialitet

Før tiltak: Sannsynlighet: 2 Konsekvens: 3 Risiko: 6
Etter tiltak: Sannsynlighet: 1 Konsekvens: 2 Risiko: 2
Spesifikke tiltak: Tofaktorautentisering, loggføring

4	
Trussel	Kriminell aktør, konkurrerende idrettsklubb eller helsepersonell
Sårbarhet	Sikkerhetsfeil ved systemet/dårlige intensjoner hos registrert helsepersonell
Handling	Aktør med tilgang til systemet gjør uriktige endringer på behandlingsplaner og/eller skader
Verdi	Behandlingsplaner/skader
Motivasjon	Sabotasje
Utfall	Tap av integritet i viktig, personlig informasjon

Før tiltak: Sannsynlighet: 1 Konsekvens: 4 Risiko: 4
Etter tiltak: Sannsynlighet: 1 Konsekvens: 2 Risiko: 2
Spesifikke tiltak: Loggføring

Konsekvens	Liten	Middels	Stor	Katastrofal
Usannsynlig				4
Noe sannsynlig			3	2
Sannsynlig			1	
Svært sannsynlig				

Tabell 6: Risikoer før tiltak

Konsekvens	Liten	Middels	Stor	Katastrofal
Usannsynlig		3, 4		2
Noe sannsynlig			1	
Sannsynlig				
Svært sannsynlig				

Tabell 7: Risikoer etter tiltak

Tiltak

- Loggføring
  - Ved utviklingen av systemet bør det bygges inn automatisk loggføring av all aktivitet. Det bør føres logg over alle på- og avlogginger og alle funksjoner i systemet, med tidspunkt og brukeren som utførte aktiviteten. Dette for lettere å kunne spore og avdekke uautorisert bruk.
  - **Strategi:** Avdekke uautorisert bruk slik at videre tiltak kan settes inn, kunne spore slik bruk under etterforskning
  - **Nytte:** 2
- Tofaktorautentisering
  - Ved utviklingen av systemet bør det bygges inn tofaktorautentisering i innloggingsprosessen. Dette må ikke være valgfritt; det bør være et krav at alle brukere av systemet kan sette opp tofaktorautentisering, og systemet må støtte en 2FA-løsning som folk flest kan bruke. Merknad: Vi kommer i appen til å implementere 2FA uavhengig av denne risikovurderingen; mer om det senere.
  - **Strategi:** Gjøre det vanskeligere for uvedkommende å logge seg inn med stjålet legitimasjon
  - **Nytte:** 3
- DDoS-beskyttelse
  - Den enkleste måten å beskytte applikasjonen mot DDoS-angrep på er nok ved hjelp av en webtjeneste som f.eks. Cloudflare. Dette er ikke relevant for dette prosjektet, men det bør tas hensyn til ved videreutvikling. En webtjeneste er sannsynligvis det beste for en webapplikasjon i oppstartsfasen, når bedriften ikke er stor nok til å bygge slik beskyttelse selv.
  - **Strategi:** Stoppe tjenestenektangrep fra å ta ned webapplikasjonen
  - **Har kostnad**
  - **Nytte:** 4

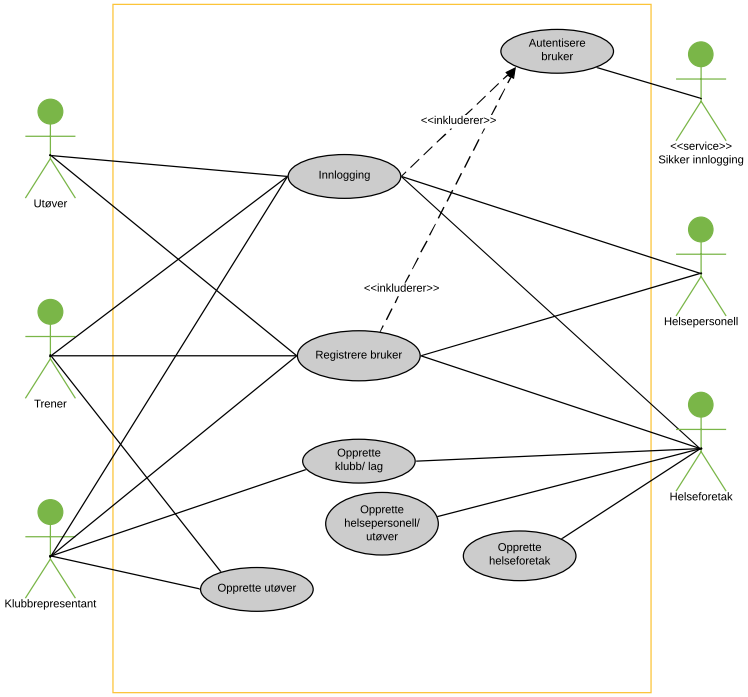
3 Kravspesifikasjon

3.1 Funksjonelle krav

Vi har valgt å bruke Use Case-diagrammer for å få et helhetlig bilde av funksjonaliteten i systemet. Vi har delt funksjonaliteten i to diagrammer; en for innlogging og registrering av brukere (figur 1), og en for funksjonalitet inne i applikasjonen (figur 2).

Det er tre hovedaktører: helsepersonell, trener og utøver. Disse representerer de vanlige brukerne av systemet. Ved oppretting av kundeforhold vil idrettsklubb og helseforetak være involvert, som vist i diagrammet for innlogging og registrering. Med aktøren helseforetak mener vi en representant for foretaket, på samme måte som en klubbrepresentant. Det brukes tjenester for sikker innlogging og pushvarslinger. Til slutt vil det være en systemadministrator som har direkte tilgang til systemet, og som derfor ikke har noen use case.

For å vise hvordan push-varslinger brukes har vi lagt ved et lavnivå use case for meldingsutveksling knyttet til en skade. Beskrivelse av hvordan dette foregår på klienten står i seksjon 6.2.3. For et helhetlig bilde vises dette scenariet i et sekvensdiagram i seksjon 6.6.



Figur 1: Use case - innlogging og registrering

Figur 2: Use case - funksjonalitet

3.1.1 Høynivå use case

Use case	Registrere bruker
Aktør	Trener, utøver, helsepersonell, helseforetak, klubbrepresentant
Mål	Registrere en brukerprofil
Beskrivelse	Denne funksjonen er ulik for vanlige brukere (trener, utøver og helsepersonell) og andre (helseforetak og klubbrepresentant). Vanlige brukere får en offentlig registreringsform, mens den andre registreringsformen er lukket og åpnes ved opprettelse av kundeforhold. Alle lager eget passord, som sjekkes for styrke og godkjennes deretter. Alle må også oppgi en invitasjonskode som gis av laget, klubben eller helseforetaket, som knytter dem til denne organisasjonen. Vanlige brukere velger om de er utøver, trener eller helsepersonell, og fyller inn brukeropplysninger som fullt navn, fødselsnummer, epostadresse og telefon.

Use case	Innlogging
Aktør	Trener, utøver, helsepersonell, helseforetak, klubbrepresentant
Mål	Logge inn på en brukerprofil for å få tilgang til systemet
Beskrivelse	Brukeren kan velge å logge på med epost eller fødselsnummer, i tillegg til å oppgi passord. Ved førstegangspålogging må brukeren sette opp tofaktorautentisering. Han bes om å scanne en QR-kode eller lime inn en tekstkode i en tofaktor-app, og så skrive inn koden generert fra denne. Ved senere innlogginger må brukeren skrive inn koden fra tofaktor-appen.

Use case	Autentisere bruker
Aktør	Tjeneste for sikker innlogging
Mål	Autentisere en pålogging med en andre faktor
Beskrivelse	Tjenesten for sikker innlogging er en tofaktor-app. Denne tar en QR-kode eller tekstkode og genererer tidsbaserte nøkler utfra koden. Når brukeren skriver inn en slik nøkkel kontrolleres den i vår app, slik at brukeren kan autentiseres. Tofaktor-appen er på mobil eller nettbrett og fungerer som en andre faktor.

Use case	**Opprette klubb/lag**
Aktør	Klubbrepresentant, helseforetak
Mål	Registrere en klubb/et lag i systemet som utøvere kan knyttes opp mot
Beskrivelse	Ved registrering av klubb fyller aktøren ut en registreringsform med informasjon om klubben, slik som navn, telefonnummer og Brønnøysund-registreringsnummer. Ved registrering av lag fyller aktøren ut en enkel registreringsform der de må oppgi lagnavn. I begge registreringsformene må man lage en unik, hemmelig invitasjonskode. Dette er en forutsetning for at utøvere og trenere kan registrere sine brukerprofiler.

Use case	**Opprette utøver**
Aktør	Klubbrepresentant, trener
Mål	Registrere en brukerprofil på vegne av en utøver
Beskrivelse	Denne funksjonen gir en alternativ måte å registrere brukere på. Her er det laget eller klubben som fyller ut en registreringsform med brukeropplysningene til en utøver, og lager et passord for dem. Når utøveren så logger inn for første gang setter han opp tofaktorautentisering, som beskrevet over.

Use case	**Opprette helseforetak**
Aktør	Helseforetak
Mål	Registrere et helseforetak i systemet som helsepersonell kan knyttes opp mot
Beskrivelse	På samme måte som oppretting av klubb kan helseforetak fylle ut en registreringsform med informasjon om foretaket, der de også må lage en invitasjonskode. Dette er en forutsetning for at helsepersonell kan registrere sine brukerprofiler.

Use case	**Opprette helsepersonell/utøver**
Aktør	Helseforetak
Mål	Registrere en brukerprofil på vegne av en helsefaglig eller en utøver
Beskrivelse	På samme måte som klubbrepresentanter over kan helseforetak fylle ut en registreringsform som oppretter en brukerprofil.

Use case	**Registrere skade**
Aktør	Utøver, helsepersonell
Mål	Registrere en ny skade
Beskrivelse	Utøveren registrerer en skade ved å fylle inn en tittel for skaden, en valgfri tekstboks for beskrivelse/ mer informasjon og krysser av/ fjerner krysset for om han vil at helsepersonell skal ta kontakt for å avtale time. Alternativt kan helsepersonell registrere skaden; dette skjer i en egen registreringsform der man fyller ut tittel, beskrivelse og dato-en skaden oppstod.

Use case	**Oversikt over skader**
Aktør	Utøver
Mål	Se oversikt over egne skader (nåværende og tidligere)
Beskrivelse	Utøveren har en liste over alle sine skader der de kan se all registrert informasjon om hver. De kan ikke slette skader, bare arkivere dem, både i tilfelle en skade kommer tilbake og for historikkens skyld. Utøveren kan gjenopprette arkiverte skader og endre tittelen på skader.

Use case	**Meldingsutveksling**
Aktør	Utøver, helsepersonell
Mål	Sikker kommunikasjon mellom utøver og helsepersonell
Beskrivelse	Hver utøver har en meldingskanal med alle sine tilknyttede helsepersonell. Her kan alle medlemmene av kanalen lese og skrive meldinger i sanntid, og kanalen er lukket og kun tilgjengelig for medlemmene. Denne kanalen er for generell kommunikasjon, i motsetning til kanaler knyttet til skader (under).

Use case	**Meldingsutveksling knyttet til en skade**
Aktør	Utøver, helsepersonell
Mål	Sikker kommunikasjon mellom utøver og helsepersonell rundt en skade
Beskrivelse	Hver utøver har i tillegg en meldingskanal per skade med alle sine tilknyttede helsepersonell. Utøveren kan benytte denne kanalen til for eksempel å gi ny informasjon om en skade. Meldingskanaler for arkiverte skader holdes tilgjengelige for historikkens skyld.

Use case	**Endre brukeropplysninger**
Aktør	Utøver, trener, helsepersonell
Mål	Administrere hva systemet lagrer om egen brukerprofil
Beskrivelse	Alle brukere av systemet kan når som helst endre brukeropplysningene fullt navn, telefonnummer og passord.

Use case	**Lagoversikt med spillerstatus**
Aktør	Trener
Mål	Få oversikt over spillerne på et lag
Beskrivelse	En tabell som viser alle utøvere på et lag. Tabellen inneholder fullt navn, spillerstatus, evt. draktnummer, evt. posisjon og evt. merknad. Den er sortert slik at spillere med skader kommer først. I tillegg er det en oversikt over antall spillere tilgjengelige, antall spillere ute og skaderate.

Use case	**Administrere behandlingsplan**
Aktør	Helsepersonell
Mål	Holde en utøvers behandlingsplan oppdatert
Beskrivelse	En utøver kan ha en behandlingsplan hvis dette er aktuelt. For utøveren er denne alltid tilgjengelig for lesing, men helsepersonell må holde denne oppdatert. Helsepersonell kan bruke denne til å gi utøveren instruksjoner i forhold til skadebehandling og -forebygging, i tillegg til at den fungerer som dokumentasjon.

Use case	**Administrere skade**
Aktør	Helsepersonell
Mål	Holde en skade oppdatert
Beskrivelse	Helsepersonell kan endre på informasjonen om en skade: tittel, beskrivelse, datoen skaden oppstod og datoen for forventet tilbakekomst. De kan også arkivere skaden eller gjenopprette den arkiverte skaden.

Use case	Oversikt over klienter
Aktør	Helsepersonell
Mål	Få oversikt over spillerne på alle tilknyttede lag
Beskrivelse	For hvert lag den helsefaglige jobber med, en tabell med alle utøverne. Denne tabellen er i utgangspunktet lik den for trenere, men helsepersonell kan også trykke på en utøver for å komme inn på en egen side for denne utøveren. Her har han tilgang til skadene og behandlingsplanen for utøveren.

Use case	Informasjon om klient
Aktør	Helsepersonell
Mål	Se og administrere skader og behandlingsplan for en utøver
Beskrivelse	Dette er utøversiden som nevnt over. Den inneholder en liste over alle utøverens skader samt behandlingsplanen, hvis den har blitt laget.

Use case	Endre spillerstatus
Aktør	Helsepersonell
Mål	Raskt og enkelt endre spillerstatusen til en utøver
Beskrivelse	Helsepersonell har autoriteten til å sette spillerstatus for en utøver, som utøveren og treneren kan se. I tabellen over utøvere kan helsepersonell velge hva statusen skal være på en skala fra 0-4, denne er beskrevet i seksjon 3.1.3.

Use case	Sende pushvarsler
Aktør	Push-leverandør
Mål	Sende pushvarsler til brukere ved viktige hendelser
Beskrivelse	Pushvarsler sendes til brukeres enheter når det er hensiktsmessig å gi snarlig beskjed om en hendelse og brukeren ikke har applikasjonen åpen. Når en utøver registrerer en ny skade sendes det pushvarsel til alle utøverens helsepersonell. Når en melding sendes, sendes det pushvarsel til alle andre medlemmer av meldingskanalen.

3.1.2 Lavnivå use case

Use case	**Meldingsutveksling knyttet til en skade**
Aktør	Utøver, helsepersonell
Mål	Sikker kommunikasjon med informasjon angående en skade på ett sted
Pre betingelser	•Brukeren er registrert som utøver eller helsepersonell •Brukeren er logget inn •Det er registrert en skade •Skaden er aktiv •Helsepersonell er knyttet opp mot laget til utøveren
Post betingelser	•Brukeren har sendt en melding
Krav	Internettilgang
Beskrivelse	**Helsepersonell:** Brukeren velger en utøver ut i fra en liste. Det kommer opp en liste over skader denne utøveren har, og brukeren velger da hvilken av skadene han vil sende melding om. **Utøver:** Brukeren velger en skade fra en liste over egne skader som er aktive. Brukeren får opp meldingshistorikk angående den valgte skaden, og kan fylle ut meldingstekstfeltet og trykke *send*. mottakeren(e) mottar push varsling for ny melding.
Detaljert hendelsesforløp	**Helsepersonell:** 1. En liste over utøvere på lag helsepersonell har avtale med vises 2. Brukeren trykker på den utøveren han vil sende melding til **Utøver og helsepersonell:** 3. En liste over utøverens aktive skader vises 4. Brukeren trykker på den skaden han vil sende melding om 5. Meldingshistorikken til skaden vises og scroller automatisk nederst til de nyeste meldingene 6. Brukeren kan fylle ut et textfelt for ny melding 7. Brukeren kan trykke *send* for å sende meldingen 8a. Den nye meldingen vises nederst på listen over meldingshistorikken for skaden 8b. En push varsling sendes til mottakeren(e) som vises på skjermen på tidligere innloggede enheter **Alternativt:** Motaker av meldingen har meldingsutvekslingen åpen og aktiv på sin enhet: 8. En pushvarsel sendes til mottakeren(e) med den nye meldingen 9. Den nye meldingen vises nederst på listen over meldingshistorikken for skaden

3.1.3 Skala for spillerstatus

I GIT brukes det en skala 0-3 for å klassifisere en spillers skadestatus. Denne tas i bruk i systemet, med et ekstra trinn for å angi spillerstatus når en ny skade er registrert, men ikke enda utredet av helsepersonell. Verdien som brukes for denne er intern og vises ikke for brukere.

0	Skadefri
1	Smertetilstand, men kampklar
2	Skade/smertetilstand, kan delta på kollektiv trening men kan måtte ha tilpasning, ikke kampklar
3	Skade/smertetilstand, kan ikke trene kollektivt og ikke kampklar
4	[Intern] Ikke utredet

Tabell 8: Skala for spillerstatus

3.2 Systemkrav

Brukervennlighet

- Brukergrensesnittet skal være enkelt, dvs. at det skal være fritt for distraksjoner og bare vise det som er nødvendig.
- Brukergrensesnittet skal ha en enkel men sportslig designprofil som er attraktiv for målgruppen.

Pålitelighet

- Applikasjonen skal gi raskest mulig respons på brukerinput og -forespørsler
- Ved krasjing eller mistet internettilkobling skal brukeren gis beskjed

Tilgjengelighet

- Alle sider skal fungere optimalt på både mobil og PC

Fleksibilitet

- Alle sidene i applikasjonen skal fungere i landskapsorientering på mobil

Interoperabilitet

- Applikasjonen skal støtte de mest populære versjonene av Chrome, Safari, Edge, Firefox og Opera på PC [5].
- På mobil skal applikasjonen støtte Chrome og Safari.

3.3 Sikkerhetskrav

Systemet skal:

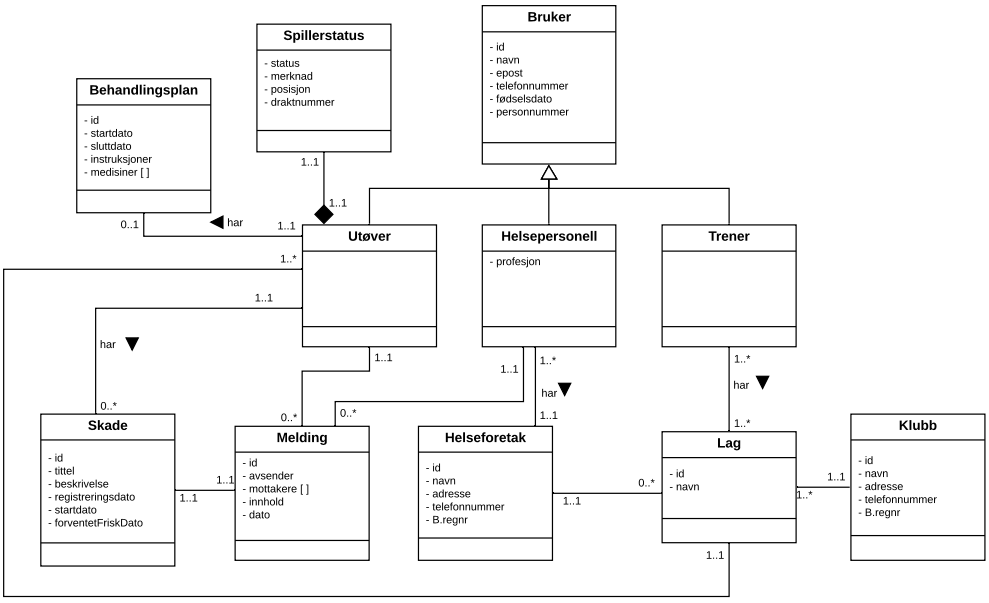
- Overholde personvernloven
- Alltid kreve tofaktorautentisering ved innlogging
- Bruke TLS/HTTPS på all kommunikasjon mellom klient og server
- Kryptere alle brukerinnskrevne passord før de lagres i databasen

3.4 Operasjonelle krav

Systemet skal:

- Loggføre all brukeraktivitet med innlogget bruker og tidspunkt
- Tåle at opptil 250 personer bruker appen samtidig uten at ytelsen går under akseptabelt nivå
- Ikke ha mer enn èn time nedetid totalt i måneden

3.5 Domenemodell



Figur 3: Domenemodell

Domenemodellen mangler klubbrepresentanter, representanter for helseforetak og systemadministrator da disse er mest relevante ved opprettelse av kundeforhold/utrulling av appen.

4 Valg av teknologier

Vi har gjort noen teknologiske valg, på bakgrunn av research og ønsker fra oppdragsgiver, for visning av applikasjonen på mobil, server-løsninger og utviklingsmiljøet. I dette kapitlet beskriver og begrunner vi disse valgene.

4.1 Applikasjon på mobil/ systemet tilgjengelig på mobil

Som nevnt i seksjon 1.2 skal vi ikke utvikle en native applikasjon for mobil, men systemet skal være tilgjengelig både på mobil og desktop. Systemet trenger ikke andre native funksjonaliteter enn push-varsling.

Vi har vurdert progressiv web-applikasjon tilpasset mobil og hybrid-applikasjoner.

4.1.1 Progressiv web-applikasjon

Progressive web-applikasjoner, heretter refert til som PWA-er, er web-applikasjoner som lastes inn som vanlige websider, men som har mulighet til å tilby brukeren funksjonaliteter som å arbeide offline, push-varslinger, feste web-appliksjonen til hjem-skjermen, splash screen, geolokasjon samt gi tilgang til mediefunskjoner som mikrofon, video- og kameraenheter. Dette har tidligere kun vært mulig med native applikasjoner. Det arbeides i skrivende stund med støtte for Bluetooth, og i framtiden vil det komme støtte for sensorer (akselerometer, gyroskop, nærhetssensor, fingeravtrykk m.m.).

De fleste vanlige nettlesere i dag støtter push-varsling ved bruk av Push API [12]. Av nettleserne vi skal dekke, nevnt i avsnitt 1.7, støtter ikke Safari Push API verken på desktop eller mobil. For desktop har Safari støtte for *Safari Push Notifications*, men dette er ikke støttet på mobil ennå [13].

PWA-er bruker blant annet teknologien service worker. Service workers kjører kun over HTTPS grunnet de vide mulighetene for angrep som ellers hadde vært åpne. HTTPS er derfor et krav for PWA-er.

4.1.2 Hybrider

En hybrid-applikasjon er en blanding av web- og native applikasjoner. Den kan installeres på mobil som en native applikasjon, men er vanligvis en web-applikasjon, og kjører i et webview [14]. Akkurat som PWA-er er disse applikasjonene ofte bygd med JavaScript. Man utvikler en web-applikasjon og bruker et rammeverk som en bro mellom plattformene for å få tilgang til bl.a. enhetens hardware.

Slike applikasjoner er enklere og billigere å lage enn native applikasjoner da de bygger på webteknologi. Man får tilgang til enhetens lagring, kamera, GPS osv.

4.1.3 Valg av løsning for tilgjengelighet på mobil

Vi valgte å gå for en PWA på grunnlag av en helhetsvurdering oppsummert i tabell 9 med forklaringer på våre vurderinger på punktene der det var forskjell på de to løsningene.

Krav	**PWA**	**Hybrid**
Kostnad	•	
Push-varsler		•
Deployment	•	
Enkelt å legge til funksjonaliteter	•	
Tilgjengelig fra hjem-skjermen	•	•
Rask	•	
Vår kunnskap	•	

Tabell 9: Sammenligning av hybrid- og progressive web-applikasjoner som løsning på mobil

Kostnader

En hybrid-løsning har høyere kostnader ved videreutvikling grunnet bruk av flere rammeverk for å kunne støtte ulike plattformer. En PWA kan nås på alle plattformer gjennom en nettleser, og det er enklere å tilpasse en nettside til forskjellige nettlesere enn det er å støtte flere plattformer i en hybrid-løsning.

Push-varsler

PWA mangler støtte for push-varslinger på iOS-enheter. Etter diskusjon med gruppen lar vi ikke dette avgjøre valget alene. Det er sannsynlig at Apple i fremtiden vil implementere støtte for dette da de allerede støtter push-varslinger i Safari på desktop.

Deployment

Hybrid applikasjoner må søkes opp i en mobil-app-distribusjonsplattform og lastes ned før de kan brukes, mens en PWA kan deles gjennom en URL, og det støttes nå også distribuering av PWA-er i *Google Play Store* [15]. Det er enklere å dele en PWA enn hybrid-applikasjoner med link-deling i tilegg til nedlastning i *Google Play Store*.

Enkelt å legge til funksjonaliteter

Vi regner med at systemet vil få mange endringer under vår utvikling og under pilotprosjektet, og vi legger derfor vekt på at det skal være enkelt å legge inn endringer og teste

disse. En PWA uten et tredjeparts rammeverk vil være enklere å endre på enn en hybrid-løsning hvor dette blir mer komplisert. PWA applikasjonen er den samme for desktop og mobil, og endringer her vises i nettleseren umiddelbart.

Rask

PWA-er og hyrider bruker lang tid på laste inn. Med PWA kan man velge hva som skal lastes inn først, for så å laste inn resten mens det ser ut som applikasjonen er klar til bruk.

Vår kunnskap

Ingen på gruppen har kunnskap om web-applikasjoner eller rammeverk for hybrid-løsninger. For å bygge en hybrid må vi lære oss både JavaScript og rammeverk for hver plattform, mens med en PWA trenger vi ikke sistnevnte.

4.1.4 Status for PWA nå og i framtiden

PWA kombinerer altså fleksibiliteten til internett med opplevelsen av en native-applikasjon. Store aktører som Twitter, Tinder og Trivago, for å nevne noen, har tatt i bruk PWA. De opplever gode resultater fordi PWA er betydelig mindre og lastes raskere enn native-applikasjoner. Fordeler med rask nettside/applikasjon inkluderer høyere plassering på Google-søk, mer trafikk, lavere bounce rate og flere gjennomførte kjøp. [16] [17] [18]

> ”Reliable – Load instantly and never show the downasaur, even in uncertain network conditions.
>
> Fast – Respond quickly to user interactions with silky smooth animations and no janky scrolling.
>
> Engaging – Feel like a natural app on the device, with an immersive user experience.”

– sitat Google Developer Team [19]

Twitter Lite

Twitter utviklet Twitter Lite Progressive App april 2017 og resultatet er 75 % økning i antall *tweets*, bounce rate ned med 25 % og antall sider besøkt per økt har økt med 65 %.

Tinder

Tinder har redusert lastetiden fra 11,91 sekunder til 4,69 sekunder med deres PWA. Den er 90 % mindre enn enn deres native Android-applikasjon.

Trivago

Trivago opplevde en 150 % økning i antall som la til deres applikasjon på hjem-skjermen, det økte engasjementet gav 97 % økning i antall som takket ja til hotelltilbud. Brukeren

kan fortsette å bruke nettsiden deres offline, og 67 % fortsatte etter at de kom online igjen.

4.2 Google Polymer

Etter råd fra faglærere og vårt ønske om å unngå rammeverk mest mulig, bestemte vi oss for lage applikasjonen med Google Polymer. Googles Polymer Project er en gruppe som arbeider med å lage verktøyer, biblioteker og standarder innen webutvikling [20]. De fronter utviklingen av blant annet PWA-konseptet, og har utgitt verktøyet *PWA Starter Kit* [21] for å bygge nye slike applikasjoner.

4.2.1 PWA Starter Kit

PWA Starter Kit er en grunnpakke som man kan bygge videre på. Den inneholder et manifest, service worker, implementasjon av PRPL-patternet, responsiv layout, 7.1.1), enkel løsning for routing og god dokumentasjon. Det lar deg utnytte det siste innen moderne webutviklingsfunksjonaliteter som JavaScript moduler, web-komponenter og HTTP/2. Mer om dette fra og med 4.2.3. Polymer anbefaler dette verktøyet for nye applikasjoner, ettersom det er bygd på de nyeste og raskest voksende konseptene innen webutvikling. [22]

Siden ingen i gruppen har erfaring med webutvikling og dermed heller ikke PWA valgte vi å ta utgangspunkt i *PWA Starter Kit* for å komme fort i gang. I tillegg har Johanne og Cecilie Polymer-biblioteket som pensum i emnet IMT2291 denne våren, og vil da ha tilgang på hjelp fra foreleser og introduksjonsmateriale.

PWA Starter Kit kommer i to utgaver; med og uten Redux.

4.2.2 Redux

Redux er et styringsverktøy for en applikasjons state. All data som hentes fra serveren kan lagres i en store som lagrer applikasjonens state-tre. Dette kan gjøre applikasjonen treg da det gjør klienten tung. Men med lazy loading henter applikasjonen data på en smartere måte, slik at første siden lastes inn først, og så hentes de andre ressursene etter hvert som de trengs.

Ut ifra domenemodellen i seksjon 3.5 ser vi at ulike komponenter i applikasjonen er avhengig av tilgang til samme data, men ikke alle komponentene har direkte forhold til hverandre. Når data i et komponent endres, må det oppdateres i alle andre komponenter som også bruker disse dataene. Ved bruk av Redux får alle komponenter tilgang til state hvor all data er lagret. Vi valgte å bruke Redux. Dette løser problemet med dataflyt, men det introduserer også en sikkerhetsrisiko når all data ligger på en sentral plass. Hvordan

vi løser dette er beskrevet i seksjon 6.2.3.

4.2.3 LitElement

LitElement er en enkel base-klasse (kun 6,6 kB) for å lage raske og lette webkomponenter. Man kan bruke Vanilla Javascript, og et LitElement-komponent oppdateres automatisk når dets *properties* forandres (4.2.3). LitElement benytter seg av lit-html (4.2.3) og følger webkomponent-standarden (4.3) [23]

I resten av 4.2.3 nevnes de egenskapene med LitElement vi har benyttet oss av. For mer informasjon henvises det til [22].

Properties

Et LitElement-komponent kan holde styr på egne *properties*, heretter refert til som egenskaper, og dets attributter. Som standard vil komponentet oppdateres når verdien til deklarerte egenskaper endrer seg. Syntaks for deklarasjon av data finnes i kodeliste 6.4 [24]

Templates

LitElement fremstiller og re-fremstiller templates asynkront under oppdatering ved forandring av deklarert data. Under en oppdatering er det kun den delen av DOM som får sine data forandret som blir re-fremstilt. Syntaks for å definere og fremstille LitElement-template, kodeliste 6.5 [25]

lit-html

lit-html er et template-bibliotek for JavaScript. Det lar deg skrive HTML templates i JavaScript ved å bruke *template-literals* med innebygde JavaScript-uttrykk. Bak scenen lager lit-element `<template>`-elementer fra JavaScript-templatene. Syntaks for å bruke lit-html er i kodeliste 6.5 [26]

Lifecycle

Komponenter som er basert på LitElement oppdateres som nevnt asynkront ved å observere forandringer i deklarerte egenskaper. LitElement arver de samme *lifecycle-callbacks* som fra webkomponent-standarden. Det referes til [27] for mer informasjon om dette. Kort fortalt er dette *callback* -funksjoner som re-fremstiller *templates* ved ulike tidspunkt-/hendelser. Vi har brukt LitElements *firstUpdated()* som påberopes etter at elementets *DOM* blir opdatert førsts gang, umiddelbart før updated() blir påberopt, og *updated()* som påberopes hver gang et elements *DOM* har blitt oppdatert og fremstilt. [28].

Styles

Som nevnt i 4.3.2 kan webcomponenter adskille og gjemme stil fra annen kode på siden. LitElement benytter seg av en css-hjelpefunksjon som lar deg definerer statisk stil som gjelder for alle instanser av LitElementet det måtte gjelde. Se 6.6.

4.3 Webkomponenter

Webkomponenter er en pakke med forskjellige teknologier som gjør det mulig å lage gjenbrukbare egendefinerte elementer - med funksjonaliteten innkapslet vekk fra resten av koden - og bruke dem i webapper. Webkomponent-standarden består av består av fire teknologier [29]:

- Custom Elements - API til å definere et nytt HTML-element
- Shadow DOM - innkapslet DOM og stilisering
- HTML Templates - HTML-fragment som ikke gjennomgår rendering, men lagres til det blir kalt på av Javascript.
- HTML Imports - foreldet og ikke anbefalt brukt. [30]

Webkomponenter ble introdusert for første gang av Alex Russell på Fronteers Conference i 2011 [31]. Han jobber som programvareingeniør hos Google og jobber blant annet med Chrome og representerer Google på TC39, komitéen som standardiserer og utvikler ECMAScript [32]. Polymer-prosjektet [20] ble startet av Google i 2013, og i oktober 2018 kom Firefox 63 med utviklerverktøy for støtte av webkomponenter. Webkomponenter er altså et relativt nytt konsept, men det finnes flere nettssamfunn for webkomponent-økosystemet. Vi har benyttet oss en del av webcomponents.org som tilbyr et grensesnitt for søk i alle eksisterende webkomponenter [33].

4.3.1 Custom Elements - egendefinerte elementer

Et av hovedtrekkene ved Webkomponent-standarden er muligheten til å lage egendefinerte elementer som innkapsler funksjonaliteten på en HTML-side, i stedet for å gjøre det med en lang, nestet gruppe av elementer som sammen gir en tilpasset sidefunksjon.

Kontrolleren av egendefinerte elementer i et webdokument er CustomElementsRegistry-objektet. Dette objektet lar deg registrere et egendefinert element på siden, returnere informasjon om hvilke egendefinerte elementer som er registrert, etc. For å registrere et egendefinert element på siden, kodeliste 4.1, bruker man metoden CustomElementsRegistry.define(). Dette tar som sine argumenter:

- En DOM-streng som representerer navnet man gir elementet. Merk at egendefiner-

te elementnavn krever minst en bindestrek og kan ikke være enkeltord.

- Et klasseobjekt som definerer elementets oppførsel.

Navnet til klasseobjektet skal tilsvare DOM-strengen slik at hvis man omgjør sistnevnte til *Pascal Case* (store forbokstaver) og fjerner bindestrekene skal man ende opp med navnet til klasseobjektet.

Kodeliste 4.1: Definere et egendefinert komponent

```
window.customElements.define('athlete-view', AthleteView) {
...
}
```

Elementet heter 'athlete-view' og dets klasseobjekt er 'AthleteView'. Dette elementet er nå registrert i CustomElementRegistry-grensesnittet og kan impletmenters i webdokumentet som en HTML-tag, kodeliste 4.2. [34]

Kodeliste 4.2: Egendefinert HTML-tag

```
...
<athlete-view></athlete-view>
...
```

4.3.2 Shadow DOM

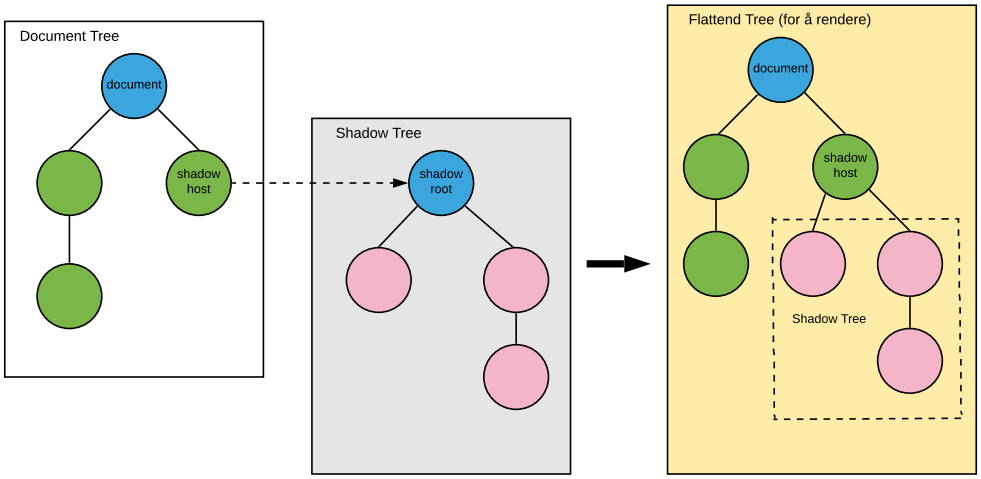
Et viktig aspekt ved webkomponenter er innkapsling, at man har muligheten til å adskille og gjemme html struktur, stil og oppførsel fra annen kode på siden. Ved å holde det separert hindrer man at ulike deler kolliderer og skaper problemer, samt at man gjør koden ryddigere. Uten denne separeringen kan kollisjoner skje for eksempel ved bruk av samme attributtnavn eller samme navn på funksjoner. Shadow DOM API-et har en nøkkelrolle i dette ved å gi mulighet til å legge til et separat, bortgjemt DOM til et element.

Shadow DOM terminologi:

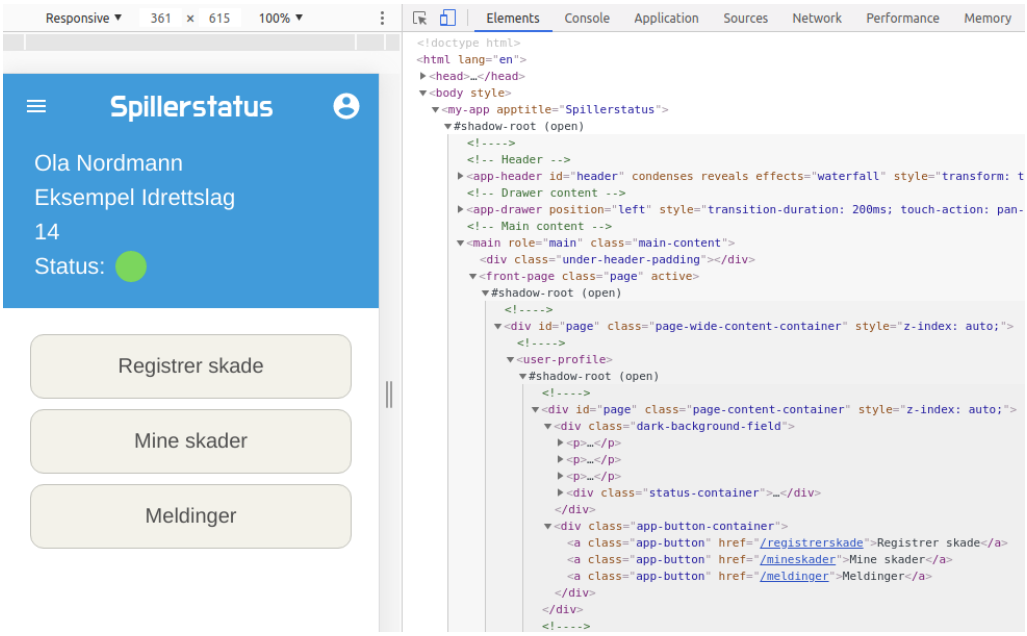
- Shadow host: den ordinære DOM-noden som shadow DOM er festet til.
- Shadow tree: DOM-treet på innsiden av shadow DOM.
- Shadow-grensen: hvor shadow DOM slutter og ordinær DOM starter.
- Shadow root: rotnoden til shadow-treet.

Man kan manipulere nodene i shadow DOM på akkurat samme måte som ikke-shadow noder. Men, kode på innsiden av shadow DOM kan ikke påvirke noe på utsiden av det [35].

I DOM-treet, figur 5, kan man se de ulike shadow DOM-elementene ligge under hver sin



Figur 4: Shadow DOM



Figur 5: Eksempel DOM-tre i applikasjonen

shadow-root, som igjen ligger under sine respektive elementer.

4.3.3 Templates

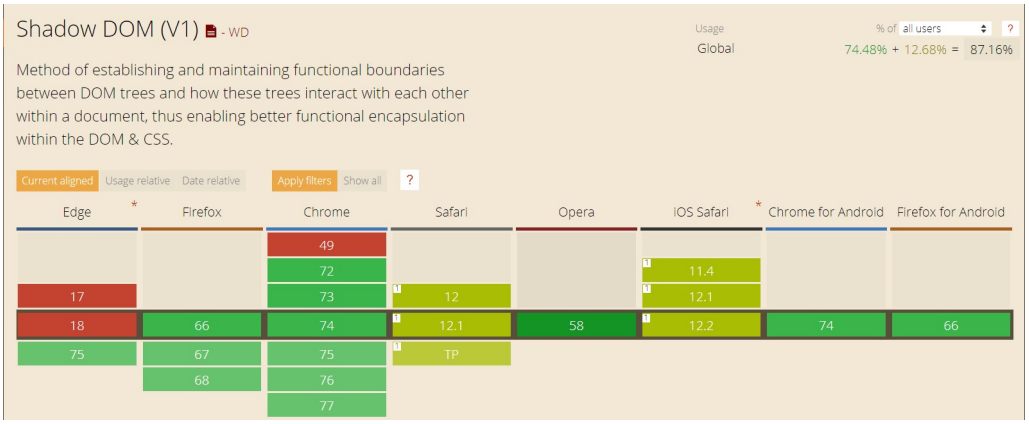
Med HTML elementet <template> kan man skrive markup-templates som ikke blir rendret i DOM, men som kan refereres til med JavaScript. Disse kan gjenbrukes og bli brukt som basis til et egendefinert elements struktur. Dette er svært nyttig når man har behov for å bruke samme markupstruktur flere ganger på en webside. `<template>`-tagen er noe vi ikke har benyttet oss av direkte, men vi lager templates i LitElement (4.2.3). Her er et trivielt eksempel på bruk av template-elementet [36]:

Kodeliste 4.3: HTML template-element

```
<template id="my-paragraph">
  <p>My paragraph</p>
</template>
```

4.3.4 Støttede nettlesere

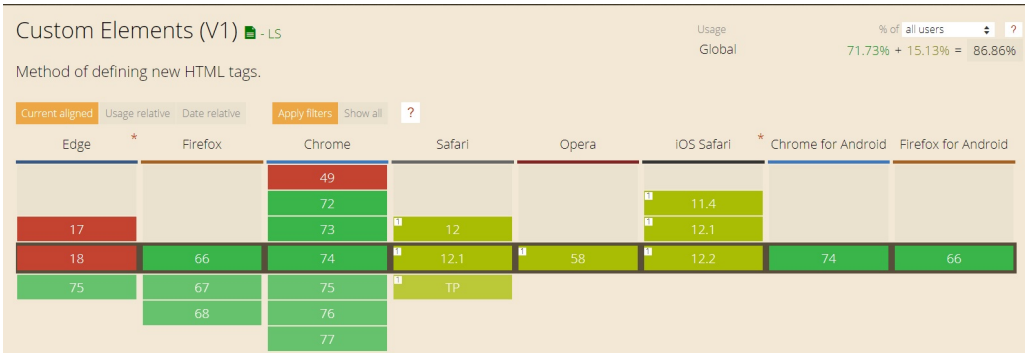
På nettsiden caniuse.com kan man søke på de teknologiene man vil bruke og se hvilke nettlesere og versjoner som støtter disse. Webkomponenter benytter seg som nevnt av Shadow DOM, Custom Elements og HTML Templates. Her er noen resultater på søk av 'webcomponents' med de nettleserne vi vil støtte, der de markert med svart ramme er siste versjon per mai 2019:



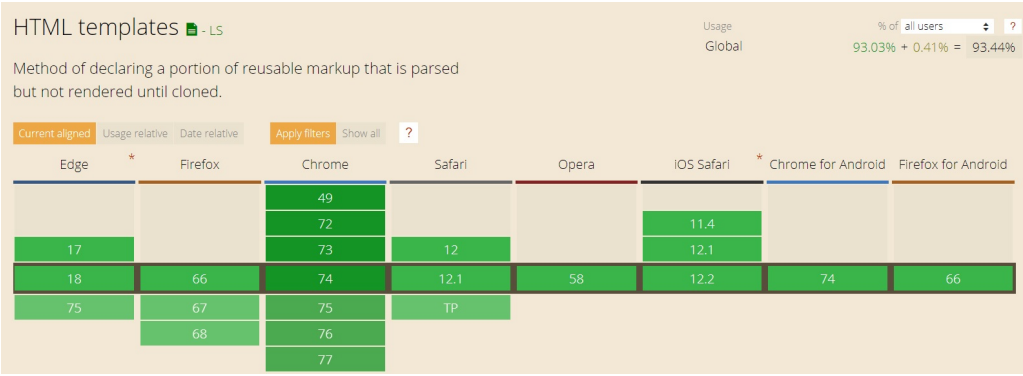
Figur 6: Nettlesere som støtter Shadow DOM v1. Skjermdump fra caniuse.com

Safari og iOS Safari har delvis støtte for Shadow DOM v1 og Custom Elements v1, og det er noe usikkert om de kommer til å støtte disse fullt ut i nærmeste framtid. I Shadow DOM v1 mangler de støtte for enkelte CSS-selektorer, som `:host > .local.child`. `::slotted()` fungerer ikke optimalt.

Chrome og Firefox, både for PC og Android, har full støtte for alle tre teknologiene, mens Edge jobber med utvikling for støtte av Shadow DOM v1 og Custom Elements v1. Alle nettleserne støtter HTML templates.



Figur 7: Nettlesere som støtter Custom Elements v1. Skjermdump fra caniuse.com



Figur 8: Nettlesere som støtter HTML templates. Skjermdump fra caniuse.com

4.3.5 PRPL

PRPL er en moderne tilnærming til webutvikling med formål om å skape raskere webapplikasjoner, spesielt for mobil. Det er et design pattern utviklet av Polymer-teamet og introdusert ved Google I/O 2016 [37], og står for:

Push Ressursene som er nødvendige for startsiden, og bare disse, pushes til klienten så tidlig som mulig.

Render Kun startsiden rendres.

Pre-cache Ressursene for resten av sidene caches på forhånd for raskere tilgang når de trengs.

Lazy-load Ressurser lastes etter hvert som de trengs (så sent som mulig).

Andre formål med PRPL er:

- Minimere *time-to-interactive* (tiden det tar før siden kan samhandles med), spesielt ved førstegangsbruk og på mobil
- Caching med maksimal effektivitet, spesielt over tid etter hvert som appen oppdateres

- Forenkle utvikling og utrulling

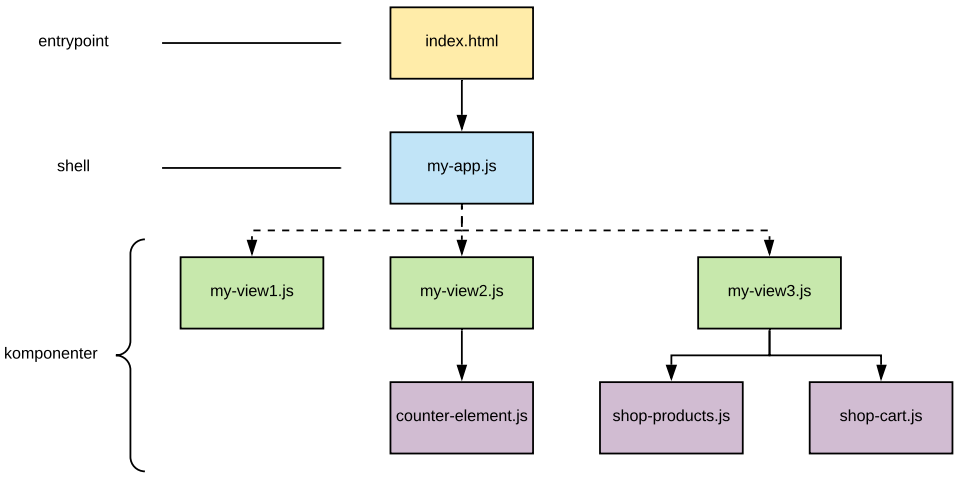
PRPL-patternet benytter seg av HTTP/2 server push (Push). Dette er en funksjonalitet i den nye teknologien HTTP/2 [38] som gjør det mulig for serveren å sende flere responser til en klientforespørsel. I tillegg til hovedresponsen kan serveren altså pushe flere ressurser til klienten, i den samme forespørselen. Dette gjør det svært effektivt å laste inn for eksempel en HTML-side med mange `<link>`- og `<script>`-tagger som angir nødvendige ressurser for siden. Med HTTP/1 måtte klienten gjøre en separat forespørsel for hver av disse ressursene.

Som nevnt er *PWA Starter Kit* bygd etter PRPL-patternet. Figur 9 viser hvordan *PWA Starter Kit* er strukturert.

Heltrukne linjer representerer statiske avhengigheter. I `index.html` vil dette si `<link>`- og `<script>`-tagger, men ellers betyr det egendefinerte tagger (webkomponenter). Stiplede linjer representerer dynamiske avhengigheter, altså web-komponenter som lastes inn etter behov.

`index.html` er appens *entrypoint*, eller inngangsport. Den importerer appens *shell*, `my-app.js`, ved hjelp av en `<script>`-tag, samt eventuelle polyfills som behøves av nettleseren. Denne filen er det første klienten forespør, og serveren pusher importerte ressurser til klienten.

Shellet inneholder den overordnede applikasjonslogikken. Den *lazy loader* komponentene etter hvert som de trengs.



Figur 9: Struktur for PWA Starter Kit

4.4 Utviklingsmiljø

4.4.1 Virtuell maskin på på SkyHiGh

Vi ønsket oss et utviklings- og testmiljø som kunne deles mellom alle tre på gruppen samt oppdragsgiver. Johanne har tidligere jobbet med virtuelle maskiner i NTNUs SkyHiGh, og vi bestemte oss for å bruke dette. Vi fikk tildelt et prosjekt og satte opp en virtuell maskin med SSH-tilgang fra våre egne maskiner. VM-en fikk en offentlig IP-adresse og vi åpnet port 80 og 443 globalt slik at oppdragsgiver kunne gå inn på denne fra utenfor NTNUs nettverk. På denne måten kunne vi blant annet demonstrere de nyeste endringene i appen på møter med oppdragsgiver.

Vi satte også opp en virtuell maskin for backup, og et backupskript som kjørte daglig i en cron-jobb og synkroniserte en kopi av prosjektmappen på backup-VM'en med prosjektmappen på utviklings-VM'en.

4.4.2 Docker Compose

Vi kjører applikasjonen i et Docker Compose-miljø med containere som kjører Docker images. Dette var en av de første avgjørelsene vi tok for applikasjonen, da Johanne hadde god erfaring med dette fra før av, i tillegg til at både Johanne og Cecilie hadde begynt å bruke dette i emnet IMT2291. Et slikt miljø er svært enkelt å replikere på utviklings-VM'en og våre egne maskiner, og vi slipper å vedlikeholde utviklingsmiljøet manuelt, slik som å oppdatere nødvendig programvare og opprettholde lik versjon for alle på gruppen. De som tar over prosjektet ved videreutvikling vil kunne starte applikasjonen umiddelbart: det eneste kravet er at man har Docker Compose installert, alt applikasjonen trenger kjører i containere.

Vi tok utgangspunkt i et eksempel fra IMT2291 med docker-compose.yml-fil og tilhørende Dockerfiler, og tilpasset det etter våre behov. Slik endte Docker Compose-miljøet vårt opp:

Container	Image	Beskrivelse
node	NodeJS	Server. Applikasjonen kjører i denne containeren. Synkroniserer hele prosjektmappen (kilden) til `/app` (i containeren) slik at endringer i koden vi gjør lokalt, automatisk reflekteres til applikasjonen.
nginx	NGINX	Brukes som reverse proxy for applikasjonen: alle HTTP-forespørsler til applikasjonen kommer først til denne containeren, mer om det i seksjon 6.3.1. Kontrollerer TLS/HTTPS, har Certbot installert.
db	MariaDB	Database. Starter med to brukere og passord for disse, som holdes skjult med Docker Secrets. Appforespørsler til databasen går til denne containeren med passord lest fra fil - aldri oppgitt i koden. Selve databasen ligger i et persistent volume som backupskriptet tar backup av. Ved oppstart leser MariaDB konfigurasjonfilene i `/mysql` og kjører i alfabetisk rekkefølge SQL-filene i `/dbInit`.
phpmyadmin	phpMyAdmin	GUI for enkel visning og manipulering av databasen. Benytter brukerne satt opp i MariaDB og henter passordene til disse. Tilgjengelig på port 8080.
sonar	SonarQube	Verktøy for statisk analyse av JavaScript-koden. På utviklings-VM'en er denne satt opp med en bruker med passord i en beskyttet fil. Tilgjengelig på port 9000.

4.4.3 TLS/HTTPS

I starten brukte appen et OpenSSL selvsignert TLS-sertifikat for HTTPS. Dette fungerte for det meste greit, men etter hvert ble det aktuelt å teste service workers. Da måtte vi ha et godkjent sertifikat, for selv om vi kunne åpne appen i Google Chrome med et ugyldig sertifikat, var dette ikke godkjent for service workers.

For å få utstedt et sertifikat måtte vi ha et domenenavn. Vi fikk tak i spillerstatus.tk gratis i 6 måneder fra Freenom. Domenenavnet er knyttet til den offentlige IP-adressen vår i SkyHiGh, som vil tas ned i løpet av sommeren 2019.

Vi fikk et gratis TLS-sertifikat fra Buypass som også utløper sommeren 2019. Dette ligger i /letsencrypt, og Certbot håndterer det i NGINX-containeren. På denne måten er all kommunikasjon mellom klient (nettleser) og server (NGINX) kryptert.

4.4.4 SonarQube

SonarQube er et verktøy for *continuous inspection* (kontinuerlig inspeksjon) av kildekode som støtter over 25 programmeringsspråk [39]. Vi valgte å ta i bruk dette ettersom Johanne og Cecilie har god erfaring med bruk av dette i IMT3281. Vi fant et Docker image for dette og installerte det i Docker Compose-miljøet vårt.

SonarQube-containeren er satt opp til å skanne JavaScript-koden i prosjektkatalogen vår. Vi kjører kommandoen `sonar-scanner` for å starte en analyse av koden på nåværende tidspunkt. I webgrensesnittet til SonarQube, tilgjengelig på port 9000, får vi så en oversikt over prosjektets kodekvalitet med bugs, sårbarheter, mindre alvorlige feil og testdekning. Vi kan gå inn på disse for å se hvor i koden feilene ligger og beskrivelser av hva som er galt med eksempler på "rett" kode. Ved å kjøre nye analyser siden kan vi sammenligne antall feil med forrige analyse og se en graf over utviklingen til prosjektet.

4.5 Server

Alle på gruppen har lært om server-side skripting i PHP i emnet IMT2571. Etter samtaler med veileder og samarbeidspartner, og på grunnlag av vår erfaring med dette språket bestemte vi oss for ikke å bruke PHP på serveren.

Etter research oppdaget vi Node.js [40], et runtime-system for server-applikasjoner som bruker JavaScript. Ettersom vi skal lage en webapplikasjon i JavaScript fant vi det naturlig å velge Node.js for server-side skripting, for vi tenkte at det ville kunne være tidsbesparende ettersom vi skulle lære oss JavaScript uansett. Dessuten ville vi slippe å forholde oss til to ulike språk og måtte bruke dem om hverandre.

PWA Starter Kit anbefaler PRPL-server [41], en enkel server som er designet etter PRPL-prinsippene, og kommer med denne innebygd. Denne serveren fungerer godt sammen med Express [42], et raskt og minimalistisk rammeverk for Node.js. Derfor tok vi utgangspunkt i PRPL-servers eksempel på et Express-serverskript for vår egen server.

4.6 Database

4.6.1 Valg av type database

Det finnes mange ulike typer databaser, der kanskje de mest kjente er relasjonsdatabase og NoSQL, eller ikke relasjons-database. Det vi la vekt på i valg av type database var i hovedsak at det skulle være enkelt for oss å sette opp og ta i bruk, samt sikker lagring av data i forhold til at vi innhenter sensitiv informasjon om brukere som fødselsnummer og helseopplysninger. SQL, relasjonsdatabaser og MariaDB/MySQL har vi alle kjennskap til gjennom et tidligere emne (IMT2571). Relasjonsdatabaser, ofte kalt SQL-databaser,

benytter seg av Structured Query Language (SQL). SQL lar deg aksessere og manipulere databaser og ble innført som standard i American National Standards Institute (ANSI) i 1986 og i International Organization for Standardization (ISO) i 1987 [43], der sistnevnte er standarden Norge benytter seg av. ISO-standarden gir trygge retningslinjer for videre utvikling av applikasjonen.

4.6.2 MariaDB

MariaDB er en videreutvikling av MySQL utviklet av programmeringsgrupper (community developed). Det er en kommersielt støttet fork av MySQL databasehåndteringssystem (RDBMS) som har som hensikt å være gratis og åpen kildekode under General Public License (GNU). MariaDB etterstreber høy kompatibilitet med MySQL, slik som eksakt samsvar med MySQL API-er og kommandoer. MariaDB benytter seg av InnoDB lagringsmotor. [44]

4.6.3 phpMyAdmin

phpMyAdmin er et gratis administrasjonsverktøy for MySQL og MariaDB med åpen kildekode. Det er skrevet hovedsaklig i PHP og er en av de mest populære [45]. phpMyAdmin kjøres direkte i nettleseren og lar deg administrere databasen gjennom et grafisk brukergrensesnitt. Alle på gruppen har brukt phpMyAdmin i emnet IMT2571.

4.6.4 Stored Procedures

En stored procedure (SP/LP (lagret prosedyre)) er forhåndslagd, i vårt tilfelle, SQL-kode som er lagret i databasen og som kan brukes om og om igjen. Disse prosedyrene kan utføres av klientapplikasjonen som om det var en egen kommando. Fordelen med slike prosedyrer, hvis man lager de på rett måte, er at klientapplikasjonen ikke trenger rettigheter for direkte SQL-kommander og det er dermed ingen mulighet for å manipulere databasen direkte ved SQL-injeksjon [46].

4.7 Andre verktøy

Gruppen har benyttet følgende verktøy på bakgrunn av erfaring ved tidligere prosjekter:

- **Bitbucket:** Prosjektet ligger i et privat Git-repository på Bitbucket. Vi arbeider på våre egne maskiner og pusher opp til dette repoet. Når vi trenger utviklings-VM'en, puller vi fra repoet ned til den.
- **Google Drive:** Alle dokumenter for research, møtereferater, spørreundersøkelsen, mockups og andre notater er lagret i en delt mappe alle på gruppen har tilgang til.
- **Overleaf:** Rapporten og prosjektplanen er skrevet i denne LaTeX-editoren.

- **Lucidchart:** Alle figurer som ikke er skjermbilder av applikasjonen eller laget i LaTeX er modellert i dette programmet. Gruppen har en delt mappe med alle figurer og diagrammer vi har laget her.
- **Dropbox:** Rapporten i Overleaf synkroniseres til Cecilies Dropbox som backup.
- **Toggl:** Vi har brukt Toggl til loggføring av arbeidstimer gjennom hele prosjektet.
- **Trello:** Vi brukte Trello til planlegging i starten, men på den tiden vi begynte å skrive kode møttes vi så ofte at vi ikke brukte dette verktøyet så mye som vi hadde tenkt.

4.8 Tredjeparts programvare

For å implementere tofaktorautentisering og push-varslinger har vi valgt å bruke tredjeparts programvare da det er langt utenfor dette prosjektets omfang og vår kunnskap å bygge dette selv.

4.8.1 Push-varslinger

Googles Firebase er en utviklingsplattform for mobil- og web-applikasjoner [47]. De tilbyr grensesnitt for flere frontend- og backend-teknologier som man kan implementere i mobil- og web-applikasjoner. Firebase er mye brukt sammen med Polymer-prosjekter, og det finnes mye dokumentasjon på de ulike tjenestene Firebase tilbyr og hvordan disse kan brukes sammen med Polymer-prosjekter [48].

Firebase Cloud Messaging [49] er en tjeneste for pushvarsling som fungerer på tvers av plattformer. Etter å ha lest om denne ser vi at den kan brukes for både pushvarsler og å sende og motta meldinger i sanntid, noe vi trenger i appen vår. Vi velger derfor å implementere *Firebase Cloud Messaging*.

4.8.2 Tofaktorautentisering

I startfasen av prosjektet anså vi BankID som den beste løsningen for tofaktorautentisering siden applikasjonen skal behandle personopplysninger. Vi tok kontakt med BankID og ble henvist til en av deres forhandlere for implementering, og da så vi at dette kom til å ta lengre tid enn hva vi hadde tilgjengelig. Sammen med dette og anbefalinger fra samarbeidspartner valgte vi å se etter en enklere løsning til pilotprosjektet.

Vi oppdaget da appen Google Autentisering, som er en gratis Android- og iOS-app som genererer nøkler for tofaktortjenester, samt NPM-pakken Speakeasy [50] som blant annet kan verifisere slike nøkler. Vi velger å bruke disse til å implementere enkel tofaktorautentisering.

5 Utviklingsprosess

I prosjektplanen, vedlagt i appendix B, la vi frem en plan for hvordan vi skulle jobbe med prosjektet. I dette kapitlet skal vi se på hvordan utviklingsprosessen har vært i forhold til planen.

5.1 Valg av utviklingsmetodikk

Systemet vi skulle utvikle hadde ikke fastbestemt størrelse eller funksjonalitet, og vi trengte derfor en smidig utviklingsmetode. Vi vurderte Scrum [51] og Kanban [52] som mulige alternativer. Scrum har en god ramme som passer bra for prosjektet vårt med tanke på endringer av systemet, men bruker mye tid på estimering og faste sprinter, noe vi ikke har mye erfaring med. Kanban er friere og åpnere, men vi fant det for åpent; vi trengte noen rammer for tidsbruk blant annet med tanke på ukentlige møter med oppdragsgiver.

Vi valgte dermed kombinasjonen Scrumban [53] som utviklingsmodell med noen tilpasninger. Daily Scrum og Scrum planning meetings ble byttet ut med en hybrid vi kalte Weekly Scrum. Weekly Scrum er ukentlige møter med Product Owner hvor vi går igjennom hva vi jobber med, veien videre og eventuelle endringer.

5.2 Gjennomføring

5.2.1 Statusmøter

I startfasen av prosjektet avtalte gruppen ukentlige møter med veileder, og videre etter behov. Gruppen avtalte å møtes fast mandag kl. 08.30 - 15.00 og fredag kl. 8.30 - 16.00, om ikke annet ble avtalt. Dette ble gjennomført, og vi har for det meste jobbet på samme rom utenom de fastsatte tidene i tillegg.

5.2.2 Weekly Scrum

Weekly Scrum med oppdragsgiver og eventuelt samarbeidspartner ble satt ukentlig, og avtaltes fortløpende. På disse møtene ble det vist hva gruppen hadde implementert, diskutert videre funksjonalitet og hva gruppen skulle jobbe med til neste gang. For hvert møte ble det skrevet møtereferat (vedlagt i appendix C) med avtalte mål for perioden frem til neste Weekly Scrum.

5.2.3 Scrumban-tavle og fremgang

I starten av prosjektet benyttet vi av oss en Trello-tavle (beskrevet i prosjektplanen B for å holde oversikt over fremgangen i prosjektet. Ettersom gruppen møttes nesten daglig og jobbet mye sammen, ble ikke tavlen like mye brukt som planlagt. Isteden ble møtereferatene brukt som en product backlog, i tillegg til god kommunikasjon i gruppen har dette gitt oversikten over status på prosjektet og fremgangen til gruppen. Etter hvert som en er ferdig med en oppgave, har gruppen avtalt hva som skal jobbes med videre og fordelt oppgaver deretter. Rapport for timelogging er vedlagt i appendix D, E og F.

Periode	
11.01 - 17.01	Til neste møte skal vi sette opp krav til systemet og skisse av funksjonaliteter i applikasjonen som er forståelig forkunden. Dette skal være mer forståelig enn typiske use-case.
18.01 - 24.01	Enringer på skissene av systemet. Use-case digrammer og kravspesifikasjon.
25.01 - 31.01	Research teknologi for applikasjon på desktop og mobil, og sikker innlogging. Jobber med utforming av spørreundersøkelse.
01.02 - 08.02	Jobber videre med spørreundersøkelsen, og jobber med risikoanalyse og krav for personvern.
09.02 - 14.02	Gjennomgang og endring av spørreundersøkelse. Sender ut spørreundersøkelse til testgruppe. Research push-varslinger.
15.02 - 21.02	Research teknologi for serverløsning. Går gjennom svar fra testrunden på spørreundersøkelsen. Finner idrettslag vi kan sende spørreundersøkelsen til. Lærer oss Polymer og Javascript.
22.02 - 01.03	Setter opp testmiljø og får på plass en første side i applikasjonen. Utsending av spørreundersøkelsen. Jobber med domenemodell og use-case. Lærer oss Polymer og Javascript.
02.03 - 07.03	Setter opp database og server. Lærer oss Polymer og Javascript.
07-03 - 14.03	Jobber med enkel innlogging i applikasjonen. Setter opp database og server.
15.03 - 21.03	Legger inn endringer i krav og use-case på bakgrunn av svar fra spørreundersøkelsen.
22.03 - 04.04	Jobber med registrering og godkjenning av utøvere og lagoversikt.
05.04 - 11.04	Jobber med å registrering av skader for utøvere. Legger inn invitasjonskode istedenfor godkjenning av utøvere ved registrering.
12.04 - 17.04	Legger inn hovedsider for trenere og helsepersonell med oversikt over utøvere og lag. Jobber med å lage inbox for skader (mellom utøver og helsepersonell, og tofaktorautentisering.
18.04 - 02.05	Push-varslinger ved nye meldinger og ny skade registrert, og tofaktorautentisering.
03.05 - 20.05	Vi prioriterer rapporten frem til innlevering. Ved siden av jobber vi med små endringer i push-varslinger for skader og meldinger, og legger inn push-varsling til trener når en spillerstatus endres.

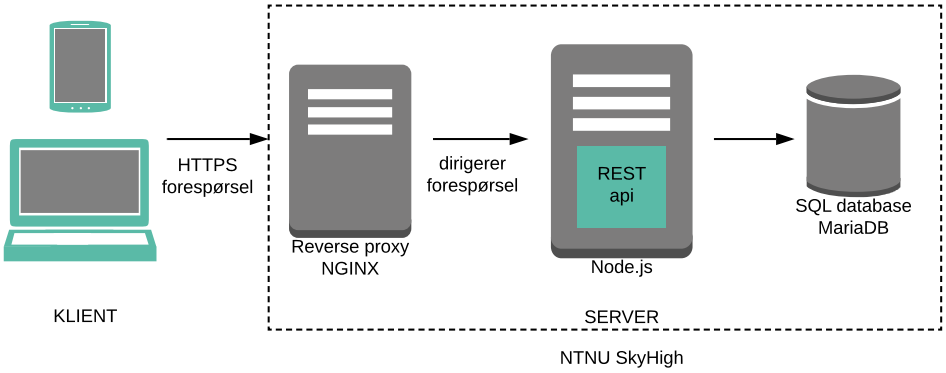
Tabell 10: Arbeidslogg mellom weekly scrums

6 Design og implementering

6.1 Overordnet struktur

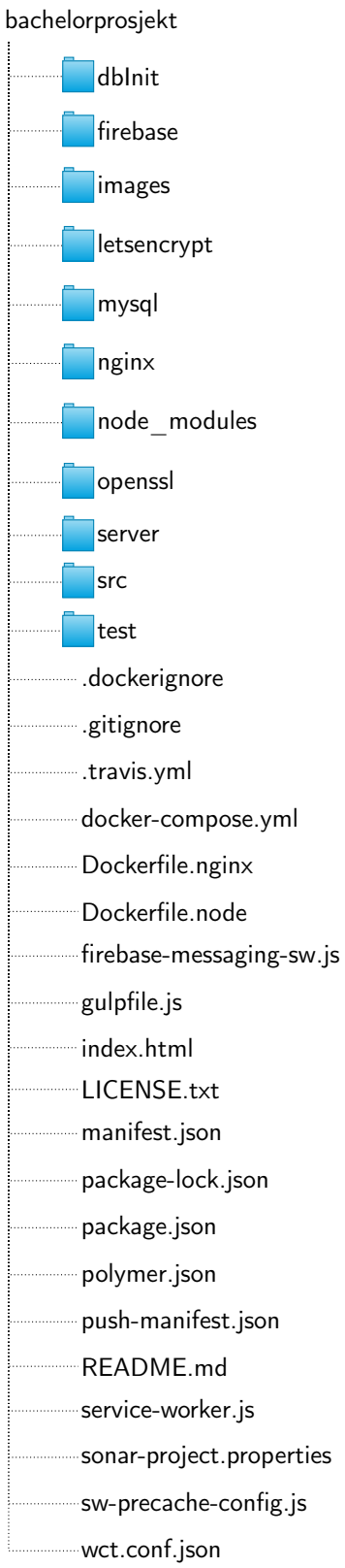
Systemet vårt har en klient-server arkitektur illustrert i figur 10. Serveren er implementert med Express rammeverk som kjører en Node.js back-end applikasjon med REST API. Forespørsler fra klienten rutes gjennom en NGINX reverse proxy til REST API-en som gjør databaseoppslag og -spørringer. Databasen er en relasjonsdatabase med MariaDB som database-server.

I tillegg til denne overordnede strukturen har vi, som nevnt i kapittel 4, brukt tredjeparts programvare for push-varslinger og tofaktorautentisering. Hvordan disse er implementer er beskrevet i 6.5 og 6.3.3



Figur 10: Oversikt over arkitekturen

All koden til prosjektet ligger i et privat repository på Bitbucket, med unntak av enkelte mapper og filer som ikke skal lastes opp. Figur 11 viser filstrukturen for det første nivået i prosjektets rotkatalog. Filene for serverlogikken ligger i `server`-katalogen, oppbyggingen av databasen i `dbInit`-katalogen, og klient-logikken i `src`-katalogen. På side 48 forklarer vi hva de ulike mappene og filene inneholder.



Figur 11: Systemets overordnede filstruktur

`firebase`: Konfigurasjonsfilen for push-varslinger med nøkler for appen vår. Lastes ikke opp til Bitbucket.

`images`: Applikasjonens favicon og startskjerm-ikon.

`letsencrypt`: TLS-sertifikatet. Lastes ikke opp til Bitbucket.

`mysql`: Konfigurasjonsfiler og passord for MariaDB. Passord lastes ikke opp til Bitbucket.

`nginx`: Konfigurasjonsfiler for NGINX.

`node_modules`: Alle NPM-pakker som appen avhenger av. Lastes ikke opp til Bitbucket.

`openssl`: OpenSSL selvsignerte sertifikater, brukes av MariaDB. Lastes ikke opp til Bitbucket.

`test`: Unit- og integrasjonstester for appen.

`.dockerignore`: Liste over filnavn Docker skal ignorere ved bygging av containere.

`.gitignore`: Liste over filnavn som ikke skal lastes opp til Bitbucket.

`.travis.yml`: Konfigurasjonsfil for tester.

`docker-compose.yml`: Konfigurasjonsfilen for Docker Compose. Definerer systemets tjenester og brukes for å sette opp et miljø som er likt på alle maskinene til utviklerteamet.

`Dockerfile.nginx` og `Dockerfile.node`: Filene inneholder kommandoer som kjøres ved bygging av containerne `nginx` og `node`.

`firebase-messaging-sw.js`: Definerer en service worker for push-varsling ved bruk av *Firebase Cloud Messaging*. Denne kjøres og legges til i klientens nettleser hvis den støtter FCM.

`index.html`: Inngangsporten for klienten; starter applikasjonen.

`LICENSE.txt`: Polymer-lisensen.

`manifest.json`: Inneholder metadata for applikasjonen som navn, farger og ikoner.

`package-lock.json` Generert av NPM ved installasjon av avhengigheter. Inneholder komplett liste over alle avhengigheter, deres avhengigheter osv. med eksakte versjoner.

`package.json`: Konfigurasjonsfil for NPM. Definerer appens avhengigheter (pakker) samt metadata og egendefinerte kommandoer.

`polymer.json`: Konfigurasjonsfil for Polymer. *PWA Starter Kit* kommer med denne.

`push-manifest.json`: Definerer for hver side i applikasjonen de skript de behøver. Denne filen brukes for å utnytte server push med HTTP/2. Mer om dette i seksjon 4.3.5.

`README.md`: Informasjon og instruksjoner rettet mot utviklerteam.

`sonar-project.properties`: Konfigurasjonsfil for *SonarQube*. Mer om dette verktøyet i seksjon 4.4.4.

`sw-precache-config.js`: Konfigurasjonsfil for appens service worker. Definerer caching av webkomponenter og fonter. Når applikasjon bygges lages `service-worker.js`-filen i `build`-katalogen på bakgrunn av hva denne filen inneholder.

`gulpfile.js`, `service-worker.js` og `wct.conf.json`: Filene er uendret fra *PWA Starter Kit*.

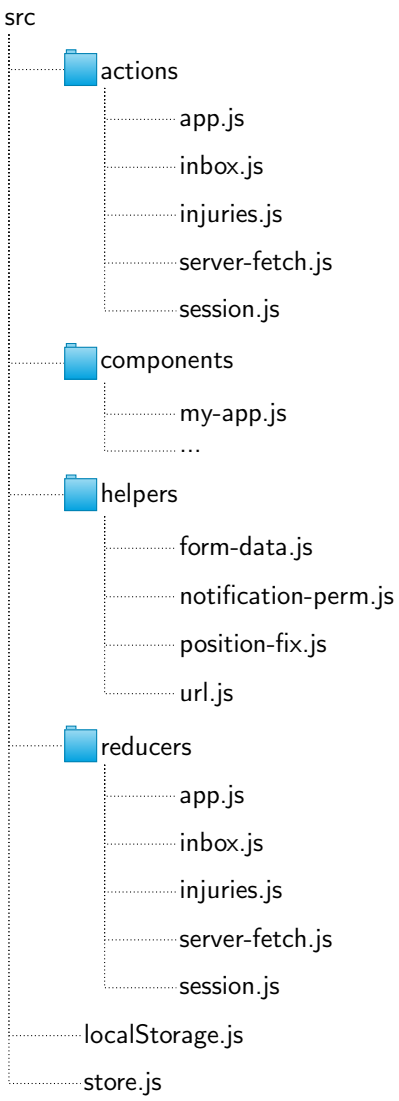
6.2 Frontend web-grensesnitt

6.2.1 Klient

Klienten er bygget på Polymers *PWA Starter Kit* med *Redux* som håndterer data. Med unntak av `index.html`-filen som ligger i rot-mappen, inneholder katalogen `src` (vist i figur 12) all kode som ligger på klienten.

Nettleseren laster inn `index.html` når en bruker åpner applikasjonen. `index.html` setter meta-data, laster inn service-workers, webkomponenter, initialiserer FCM og laster inn applikasjons-shellet, komponenten `<my-app>`, fra katalogen `components`.

Komponentene i katalogen `components` innholder klasser som defineres som web-komponenter med data og metoder for rendring av html-sider. Disse beskrives i detalj i seksjon 6.2.2.



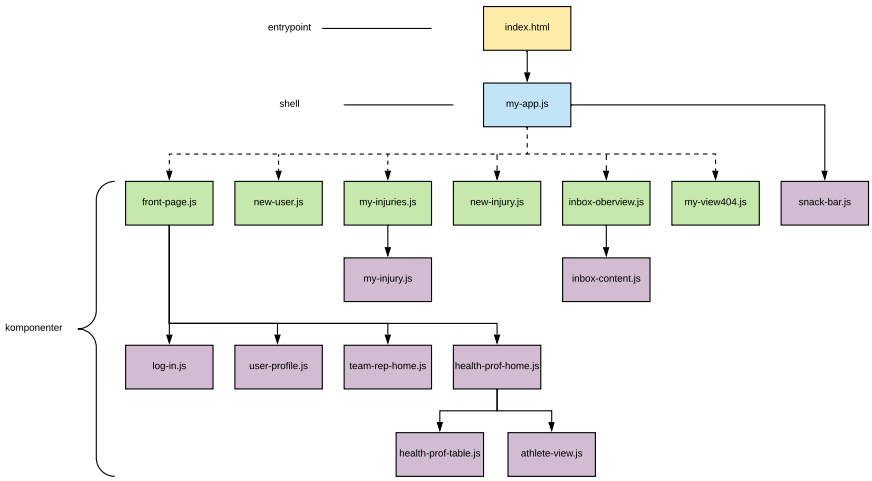
Figur 12: Oversikt over filstrukturen i mappen src

I vårt system er det flere komponenter, hvor flere er avhengig av samme data, men ikke alle komponentene har direkte forhold til hverandre. Noen innkapsler andre, men de fleste er helt separert. Når noen data i et komponent endres, må de oppdateres i de andre komponentene som også bruker disse dataene. I systemet lagres for eksempel data om utøvere, skader og meldinger. Disse har direkte sammenheng i meldings-komponentene `<inbox-overview>` og `<inbox-content>`, men data om utøvere og skader brukes også i andre komponenter som `<spiller-oversikt>` og `<athlete-view>`. For styring av dataflyt brukes Redux. All kode for håndtering av dataflyt ligger i katalogene `actions`, `reducers`, og er forklart under Redux i seksjon 6.2.3.

Klienten sender forespørsler angående data til serveren gjennom en API-forespørsel. URLen til serveren er lagret i `helpers` sammen med andre hjelpemetoder flere av komponentene bruker. Kommunikasjonen mellom klienter, server og databasen er beskrevet i seksjon 6.6

6.2.2 Webkomponenter

I katalogen `components` ligger alle komponentene i applikasjonen, disse er strukturert i et applikasjons-shell. Applikasjons-shellet laster inn de seks grønne komponentene etter behov, og ni komponenter er brukt i disse.

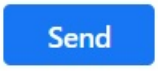


Figur 13: Struktur for web-grensesnittet

Mange ferdiglagde webkomponenter kan enkelt tas i bruk, som beskrevet i 7.1, ved å bruke dets Custom Elements HTML-tag. For eksempel Vaadin Button, figur 14, implementeres direkte med dette.

Kodeliste 6.1: HTML-kode for å legge til Vaadin Button HTML-custom element

```
<vaadin-button id="send-button"
theme="primary small">Send</vaadin-button>
```



Figur 14: Vaadin Button

Webkomponenter som har innebygd interaksjon og/eller avhenger av tilførsel av data krever i tillegg JavaScript kode. Vi skal se nærmere på Vaadin Date Picker.

Vaadin Date Picker

For å vise norsk kalender; ukedagstart mandag, norsk månedsnavn, se figur 15.

```
// Customizing vaadin-date-picker to norwegian:
const datePicker = this.shadowRoot.querySelector('vaadin-date-picker#date-picker');

datePicker.i18n = {
  week: 'uke',
  calendar: 'kalender',
  clear: 'tøm',
  today: 'i dag',
  cancel: 'avbryt',
  firstDayOfWeek: 1,
  monthNames:
  'januar_februar_mars_april_mai_juni_juli_august_september_oktober_november_desember'.split('_'),
  weekdays: 'søndag_mandag_tirsdag_onsdag_torsdag_fredag_lørdag_søndag'.split('_'),
  weekdaysShort: 'sø_ma_ti_on_to_fr_lø'.split('_'),
```

Figur 15: JavaScript for tilpasning av Vaadin Date Picker 1

i18n-objektet til `Vaadin Date Picker` modifiseres til å vise norske månedsnavn, mandag som første dag i uken, etc.

```
// Formats picked date to norwegian:
//'date' is an object -> { day: ..., month: ..., year: ... }
// need to format object to string in desired format. Here: DD.MM.YYY format:
// Note: The argument month is 0-based. This means that January = 0 and December = 11.
formatDate: function(date) {
  // to string
  date = JSON.stringify(date);
  // to array
  let dateSplit = date.split(',');
  // seperate day, month and year from array
  let day = dateSplit[0];
  let month = dateSplit[1];
  let year = dateSplit[2];
  // delete everything except digits
  day = day.replace(/[\D]+/g, '');
  month = month.replace(/[\D]+/g, '');
  year = year.replace(/[\D]+/g, '');
  // increase value of 'month' by 1. See Note.
  month = (+month+1)+'';
  // add '0' to day and month if one digit
  if (day.length == 1) day = '0' + day;
  if (month.length == 1) month = '0' + month;
  // adding '.' to string -> DD.MM.YYYY
  let fullDateFormat = day + '.' + month + '.' + year;
  return fullDateFormat;
},
// Calendar title, i.e mai 2019
formatTitle: function(monthName, fullYear) {
  return monthName + ' ' + fullYear;
}
```

Figur 16: JavaScript for tilpasning av Vaadin Date Picker 2

Videre trengs det modifisering for visning av dato på formen dd.mm.åååå. Figur 16 `JSON.stringify` for å gjøre om `date`-objektet fra Vaadin Date Picker til en string, separere dag, måned år i hver sin string, fjerning av '/' med bruk av et regulært uttrykk, legge til 0 hvis dag/dato har ett siffer og til slutt legge til '.' mellom dag, måned og år.

Overskrift i kalenderen, måned og år, legges til med `formatTitle`.

LitElement

Webkomponentene som det refereres til i 6.2.2 arver alle fra LitElement. LitElements egenskaper er beskrevet i 4.2.3. Slik kan man lage et LitElement-komponent:

1. Installere LitElement med NPM: `npm install lit-element save`
2. Importere LitElement i hvert JavaScript-dokument som skal baseres på LitElement. Kodeliste 6.2
3. Lage en klasse som skal inneholde koden for komponentet. Den arver fra LitElement. Kodeliste 6.3
4. Deklarere eventuelle egenskaper (*properties*). Kodeliste 6.4
5. Definere komponentets template med `render()`-funksjonen. Bruker import av `html`. HTML-teksten starter og slutter med ‘. Kodeliste 6.5
6. Definisjon av stil. Bruker import av `css`. CSS-koden starter og slutter med ‘. Kodeliste 6.6
7. Registrere komponentet via *customElements.define()*, se 4.3.1. Kodeliste 6.7

Kodeliste 6.2: Importering av LitElement med html og css

```
import { html, css, LitElement } from 'lit-element';
```

Kodeliste 6.3: Klasse som representer et komponent

```
class AthleteItem extends LitElement {
}
```

Kodeliste 6.4: Deklarasjon av data (*properties*)

```
static get properties() {
    return {
      firstname: { type: String },
      email: { type: String },
      lastname: { type: String }
    }
  }
```

Kodeliste 6.5: Definisjon av template

```
render() {
    return html`
          <div class="user-name">
            ${this.firstname} ${this.lastname}
          </div>
          <div class="user-email">
            ${this.email}
          </div>
    `;
```

Kodeliste 6.6: Definisjon av stil med CSS

```
static get styles() {
    return css`
        :host {
          display: flex;
          flex-direction: column;
        }
    `;
```

Kodeliste 6.7: Registrere komponenten

```
window.customElements.define('athlete-item', AthleteItem);
```

Dette resulterer i LitElement-komponentet `<athlete-item>`, kodeliste 6.8. Merk at komponentet er registrert med en bindestrek i navnet, se 4.3.1. I applikasjonen har vi en JavaScript-fil med CSS-stil som omfatter mange av komponentene vi har laget. `SharedStyles` (felles gjenbrukbare stiler) importeres og settes inn før eventuell lokal CSS om ønskelig. Kodeliste 6.9. I `render()`-funksjonen, som beskrevet i 4.2.3, brukes JavaScript *template literals* for å få tilgang til komponentets egenskaper.

Kodeliste 6.8: `<athlete-item>` LitElement-komponent

```
import { LitElement, html, css } from 'lit-element';
import { SharedStyles } from './shared-styles.js';

class AthleteItem extends LitElement {

  static get properties() {
    return {
      firstname: { type: String },
      email: { type: String },
```

```
      lastname: { type: String }
    }
  }


  static get styles() {
    return [
      SharedStyles,
      css`
        :host {
          display: flex;
          flex-direction: column;
        }
        .user-choice {
          display: flex;
          flex-direction: row;
          align-items: center;
        }
        .user-email {
          color: var(--dark-sand-color);
        }
      `
    ];
  }

  render() {
    return html`

          <div class="user-name">
            ${this.firstname} ${this.lastname}
          </div>
          <div class="user-email">
            ${this.email}
          </div>
    `;
  }
}

window.customElements.define('athlete-item', AthleteItem);
```

Kodeliste 6.9: SharedStyles

```
import { css } from 'lit-element';

export const SharedStyles = css`
  /* GENERAL */
  :host {
    display: block;
    box-sizing: border-box;
    ...
`
```

6.2.3 Redux

Redux holder styr på datalagring og -flyten på klienten. Hovedelene av Redux består av components, actions, reducers, store og state. Components inneholder alle komponentene i applikasjonen, listet i figur 13. Store lagrer applikasjonens state, mens actions og reducers brukes for å endre på applikasjonens state.

State

Singleton pattern-et [54] brukes av Redux for å lagre den eneste ene sannhet; state. Observer pattern-et [55] brukes av Redux for å la komponentene være observerere til endringer i state. State inneholder alle data brukerne ser i applikasjonen. Når state endres, oppdateres komponentene knyttet til de aktuelle delene av state.

Store lagrer state i nettleserens `local storage`. I starten av prosjektet ble data i nettleseren lagret i kompliserte objekter, et eksempel på dette er vist i kodeliste 17a. Alle data om en utøver ble lagret i et objekt. Vi endret dette etter vi fikk problemer med å legge inn endringer i state. Redux anbefaler å normalisere dataene [56].

```
athletes: {
    1: {
        ...
        injuries: {
            1: {
                ...
            },
            3: {
                ...
            }
        },
        messages: {
            5: {
                ...
            },
            8: {
                ...
            }
        }
    },
    2: {...},
    3: {...}
}
```

(a) Før

```
athletes: {
    1: {
        ...
        injuries: { 1: 1, 3: 3}
        messages: { 5: 5, 8: 8}
    },
    2: {...},
    3: {...}
},
injuries: {
    1: {
        ...
        messages: { 12: 12, 15: 15}
    },
    2: {...},
    3: {...}
},
messages: {
    5: {...},
    8: {...},
    12: {...},
    15: {...}
}
```

(b) Etter

Figur 17: Før og etter normalisering av state

Figur 17b viser dataene etter normalisering. Alle objekter av datatypen `athlete` er lagret i objektet `athletes` med `key` lik utøverens id. Hvert `athlete`-objekt har et `messages`- og `injuries`-objekt med referanser til meldinger angående utøveren og utøverens skader lagret med `key` og verdi lik meldingen og skadens id. Hvert `injury`- og `message`-objekt er lagret i objektene `injuries` og `messages` på samme måte. Hvert `injury`-objekt inneholder et `messages`-objekt med referanser til meldinger angående skaden lagret med `key`

og verdi lik meldingens id.

Dataflyt med Redux

Med Redux har alle komponenter som abonnerer på `store` dataene tilgjengelig. Når en komponent endrer noen data, oppdateres komponentene gjennom `stateChanged`-funksjonen som er innebygd i Redux. Siden alle komponenter kan få tilgang til all data som er lagret i state, kan det være en sikkerhetsrisiko. Alle dataene om utøvere, skader og meldinger slettes fra state når en bruker logger seg ut av systemet.

I tillegg til dette vil bare komponentene oppdatere seg; siden vil ikke rendres på nytt. Dette vil gi brukeren en større følelse av å bruke en applikasjon, enn en vanlig webside.

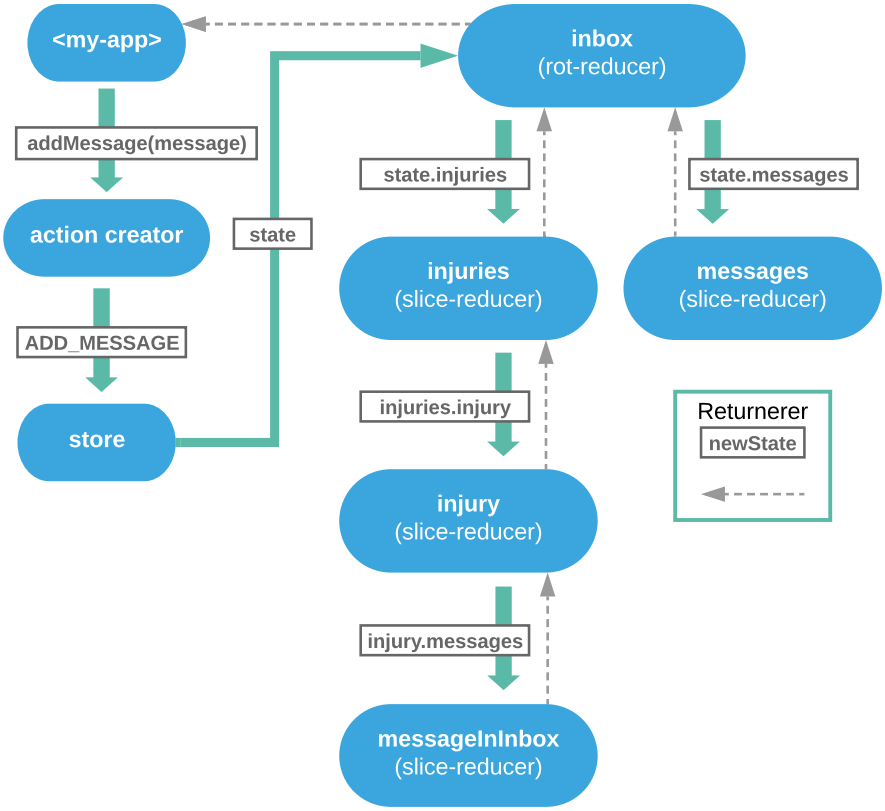
Hver endring i applikasjonen skjer via en dispatch til en action. En action videresender hva som skal endres sammen med ny data til en rot-reducer. Rot-reduceren deler noen ganger ut oppgaver videre til andre reducere (slice-reducer), eller gjør jobben selv. Alle reducerne får en kopi av sin gitte del av state og legger inn endringen i kopien før de sender den tilbake til rot-reduceren, som sender ny state tilbake til `store`. Systemet har fem rot-reducere: app, inbox, injuries, server-fetch og sessions. Disse er kort beskrevet i tabell 11. Rot-reduceren inbox er beskrevet i detalj i seksjon 6.2.3 - *Rot-reducer inbox*.

Rot-reducer	Datahåndtering
app	Kommer med *PWA Starter Kit*. Styrer hovedsidene i applikasjonen.
inbox	Gjør endringer relatert til inbox angående skader og utøvere. Denne reduceren legger inn alle meldinger, skader og utøvere, uleste meldinger og antall uleste meldinger.
injuries	Gjør endringer relatert til skader en utøver har. Denne reduceren legger inn endringer i aktive og arkiverte skader.
server-fetch	Fetch ressurser fra server. Gjør HTTP-forespørsler til API-endepunktene. Brukes for å hente ut og legge inn data på serveren.
sessions	Håndterer inn- og utlogging, brukerens id og type (helsepersonell, trener eller utøver).

Tabell 11: Oversikt over rot-reducere i systemet

Rot-reducer inbox

Figur 18 viser dataflyten når en ny melding angående en skade legges inn i state gjennom rot-reduceren `inbox`. Hvis FCM er støttet av nettleseren, mottar komponenten `<my-app>` en push-varsling og data fra FCM (detaljer om push-varslinger er beskrevet i seksjon 6.5). `<my-app>` sender en dispatch til `store` med en action lik (`ADD_MESSAGE`) og meldingen mottatt fra FCM. Action-en og meldingen sendes som et objekt laget av action creator `addMessage()` (vist i kodeliste 6.10).



Figur 18: Dataflyt når en ny melding legges til i en skade i applikasjonen

Kodeliste 6.10: Action-creator i actions/inbox.js

```
export const addMessage = (message) => {
  return {
    type: ADD_MESSAGE,
    message
  }
};
```

`store` sender en kopi av `state` til rot-reduceren `inbox` (kodeliste 6.11). Rot-reduceren sender kopi av `state.injuries`, som inneholder alle skader lagret i state, til slice-reducer `injuries`, og kopi av `state.messages` til slice-reducer `messages`.

Kodeliste 6.11: Rot-reducer: inbox

```
// Root reducer
const inbox = (state = INITIAL_STATE, action) => {
  switch(action.type){
    case ADD_MESSAGE:
      return {
        ...state,
        injuries: injuries(state.injuries, action),
        messages: messages(state.messages, action)
      };
    case ...
    default:
      return state;
  }
};
```

Slice-reduceren `messages` (kodeliste 6.12) legger inn den nye meldingen som et objekt med `key` lik meldingens id og innholdet lik objektet `message` mottatt av FCM.

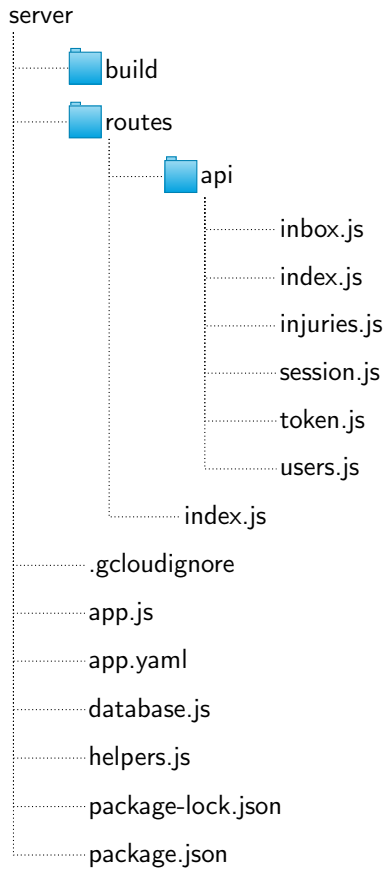
Slice-reduceren `injuries` sender skade-objektet, som meldingen tilhører, til slice-reducer `injury`. `injury` sender `injury.messages` objektet til slice-reducer `messageInInbox`. `messageInInbox` legger inn den nye meldingens id som et objekt, med `key` lik meldingens id og verdi lik meldingens id (`id:id`), inn i `injury.messages`.

Kodeliste 6.12: Slice-reducer: messages

```
// Slice reducer: it only reduces the bit of
// the state it's concerned about.
const messages = (state, action) => {
  switch (action.type) {
    case ADD_MESSAGE:
      const messageId = action.message.id;
      return {
        ...state,
        [messageId]: action.message
      };
    default:
      return state;
  }
};
```

Etter alle slice-reducere har gjort sine endringer av kopien de mottok, returnerer de kopien til reduceren de fikk kopien av. Rot-reduceren returnerer til slutt kopien av state til `store`. Alle komponenter som observerer på denne delen av `state` kjører sin egen `stateChanged(state)` funksjon. Gjennom denne funksjonen oppdateres variablene til hver komponent.

6.3 Server



Figur 19: Oversikt over filstrukturen på serveren

`build`: Den ferdigbygde applikasjonen som serves.

`routes`: Mappe for API-rutene til appen.

`api`: Mappe for rutene i `/api`.

`inbox.js`: Rutene i `/api/inbox`.

`injuries.js`: Rutene i `/api/injuries`.

`session.js`: Rutene i `/api/session`.

`token.js`: Rutene i `/api/token`.

`users.js`: Rutene i `/api/users`.

`routes/index.js og routes/api/index.js`: Filer som organiserer rutene.

`app.js`: Serverskriptet. Kjører serveren som lytter etter HTTP-tilkoblinger og server applikasjonen.

`database.js`: Skriptet for tilkobling til databasen. Kontakter `db`-containeren. API-en henter tilkoblingsobjekter herfra.

`helpers.js`: Gjenbrukbare fellesfunksjoner for API-en.

`package-lock.json` Generert av NPM ved installasjon av avhengigheter. Inneholder komplett liste over alle avhengigheter, deres avhengigheter osv. med eksakte versjoner.

`package.json`: Konfigurasjonsfil for NPM. Definerer serverens avhengigheter.

Serveren er bygget i Node.js og Express. Figur 19 viser filstrukturen.

Databaseskriptet `database.js` kontakter databasen som en MariaDB-bruker kalt `user`. Denne brukeren er satt opp i Docker Compose og brukes som databaseklient.

For å kjøre applikasjonen må den først bygges med `npm run build:prpl`. Denne kommandoen er definert i prosjektkatalogens `polymer.json`, som også definerer applikasjonens `builds`. Disse `build`-ene er ulike versjoner av applikasjonen som serves tilpasset ulike nettlesere. Polymer CLI velger blant disse når den skal serve applikasjonen, avhengig av ECMAScript-kapabilitetene til klientens nettleser.

Når applikasjonen er bygd legges ferdige `builds` i serverens `build`-katalog. Da kan den kjøres med `npm run serve:prpl`, som kjører skriptet `app.js`. Applikasjonen er da tilgjengelig på port 8081 i `node`-containeren. Men den er ikke tilgjengelig fra utsiden, det er NGINX sin jobb.

6.3.1 NGINX som reverse proxy

Vi bruker NGINX som en reverse proxy rundt applikasjonen. Docker Compose-miljøet er satt opp slik at HTTP-forespørsler til websiden mottas av NGINX-containeren, som er konfigurert til å sende disse forespørslene videre til Node-containeren, hvor applikasjonen ligger. NGINX henter responsen fra Node og returnerer den til klienten.

I Docker Compose-miljøet ligger containerne i et virtuelt nettverk slik at de kan kommunisere med hverandre. Kommunikasjonen i dette lokale nettverket er ikke kryptert, for dette skjer mellom klienten og NGINX. Altså, websiden er HTTPS-beskyttet gjennom at NGINX krypterer kommunikasjonen med klienten.

NGINX er en kraftfull motor med funksjonaliteter for å fylle mange roller, og vi bruker bare noen få av dem. En funksjonalitet vi ikke har implementert er lastbalansering. Det har ikke vært relevant i dette prosjektet siden vi utvikler en prototype, men hvis det skulle være aktuelt i fremtiden så er det relativt enkelt å sette opp dette i NGINX-konfigurasjonen.

En annen svært nyttig funksjonalitet er caching. NGINX kan mellomlagre ofte etterspurt innhold for å effektivisere trafikkflyten mellom klient og server — raske og effektive webapplikasjoner er jo et av PWA-prinsippene. NGINX kan også komprimere data for å bidra til dette.

En av fordelene med reverse proxies er at de fungerer som et ekstra beskyttelseslag rundt applikasjonen. At NGINX fanger opp klientforespørsler betyr at vi kan gjøre hva vi vil med dem. Ekstra sikkerhetstiltak kan implementeres på reverse proxyen, og alt dette er samlet på ett sted, i bare noen få konfigurasjonsfiler.

6.3.2 REST API

Vi har bygd en RESTful API med Express. Kort sagt bruker en REST API HTTP-forespørsler til å utføre GET-, POST-, PUT- og DELETE-operasjoner på data. I SQL (databasen) tilsvarer disse SELECT, INSERT, UPDATE og DELETE, og endepunktene er gruppert etter dette.

For et eksempel, anta at vi har et endepunkt `users`. For å legge til en ny bruker vil vi da gjøre en POST-forespørsel til `/api/users` med informasjon om brukeren i *request body*en, dataene som sendes med forespørselen. Eller for å hente ut informasjon om en bruker gjør vi en GET-forespørsel til `/api/users/id`, der `id` er brukerens ID i databasen.

I figur 20 beskriver vi endepunktene i vår API.

Figur 20: Oversikt over API-endepunktene

`inbox.js`: Meldingsutveksling. Sende meldinger, hente alle meldinger i en meldingskanal.

`injuries.js`: Skader. Registrere og endre skader samt hente alle skader for en utøver.

`session.js`: Innlogging. Verifisering av epost/personnummer og passord samt tofaktorautentisering.

`token.js`: Instans-ID'er for Firebase Cloud Messaging. Lagre en ny instans-ID for en bruker. Se seksjon 6.5.

`users.js`: Brukere. Registrere en ny bruker, oppdatere spillerstatus, hente informasjon om utøvere på et lag, hente informasjon om en enkelt utøver.

6.3.3 Tofaktorautentisering

Når en bruker logger inn med epost/personnummer og passord for første gang, må han sette opp tofaktorautentisering. På serveren bruker vi da Speakeasy til å generere en hemmelig nøkkel som skal lagres for denne brukeren. Nøkkelen vises som både en QR-kode og en tekstkode, og brukeren blir bedt om å legge denne koden inn i Google Autentisering eller en annen tofaktorapp ved å skanne QR-koden eller lime inn tekstkoden. Google Autentisering har en innebygd QR-skanner. Tofaktorappen må kunne generere TOTP-er (tidsbaserte engangsnøkler) fra denne koden.

Til slutt må brukeren skrive inn den gjeldende TOTP-en fra tofaktorappen. Denne oppdateres hvert 30. sekund. Når han sender inn skjemaet sjekker vi ved hjelp av Speakeasy denne nøkkelen mot brukerens kode, og hvis den er riktig logges brukeren inn.

Etter dette er det to steg for å logge seg inn: det første med epost/personnummer og passord, det andre ved å skrive inn TOTP-en fra tofaktorappen. Det geniale med TOTP-algoritmen er at appen vår ikke trenger å kommunisere med tofaktorappen; så lenge klokkene deres er synkroniserte, vil de alltid få samme nøkkel fra den samme koden.

Den andre faktoren i denne løsningen er altså en tofaktorapp som brukeren har enten på mobil eller nettbrett. Dersom brukeren ikke har noen form for skjermlås på denne enheten, vil sikkerhetseffekten av tofaktorautentiseringen dermed svekkes kraftig. Uansett vil en angriper trenge både brukerens passord og mobil/nettbrett for å logge inn på deres konto.

Vi mener dette er en god nok implementering i alle fall for en prototype. Fra et brukerperspektiv tror vi at det kunne vært mer lettvint, og kanskje en implementering av BankID eller BankID på mobil kunne ha gjort dette. Men det faller utenfor dette prosjektet og kan heller vurderes i framtiden.

6.3.4 Passordsikring

Det er ikke så mye poeng med tofaktorautentisering dersom en av faktorene lett kan passeres av en angriper. Derfor er det like viktig å implementere god passordsikring som det andre. Her beskriver vi hvordan vi har gjort dette.

Passordfeltene i både registrerings- og innloggingsskjemaene bruker selvsagt `<input type=password»`. Dette er innebygd i HTML og gjør at passordet brukeren skriver inn skjules bak prikker.

I registreringsskjemaet, der brukeren skal lage sitt passord, gir vi følgende hint: ”Passordet må være på minst 10 tegn og må ikke være et vanlig brukt passord. Bruk gjerne

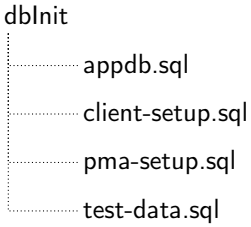
en setning, det er lett å huske men vanskelig å gjette." Vi krever altså at alle passord er minst 10 tegn lange, og dette kontrolleres på klientsiden slik at brukeren ikke kan sende inn skjemaet hvis passordet er for kort. I tillegg forsøker vi å hjelpe brukerne til å lage sterkere passord med tipset om å bruke setninger, eller `passphrases`. Dette er ett av rådene for NISTs veiledning for passordregler [57] som vi forsøker å forholde oss til.

Når brukeren så sender inn skjemaet til serveren, analyserer vi passordet med NPM-pakken zxcvbn [58]. Dette er et verktøy som sjekker strenger mot en database på over 30 000 vanlige passord i USA (det finnes nok ikke et tilsvarende verktøy for norske passord). Den sjekker også for mønstre, gjentakelse og l33t speak, og returnerer informasjon om hvor sterkt passordet er. Det setter passordet på en styrkeskala fra 0-4, der 0 er svakest og 4 er sterkest. På serveren diskvalifiserer vi passord som ligger på 0 eller 1, og forteller brukeren at passordet er for svakt. zxcvbn gir også tilbakemelding på hva som er galt med passordet, men dette er på engelsk, så vi har valgt å ikke bruke det.

Når vi har et godkjent passord, bruker vi så den innebygde Node.js-modulen crypto [59] for å hashe passordet. Først genererer vi et salt, en tilfeldig streng på 16 byte. Passordet legges til på slutten av denne strengen, og vi hasher den resulterende strengen med Password-Based Key Derivation Function 2. Til slutt legger vi til saltet på starten av denne strengen med et unikt tegn i midten slik at vi kan splitte dem senere. Den resulterende strengen, saltet og hashet, kan nå lagres i databasen.

Når brukeren sender inn innloggingsskjemaet, må vi først hente saltet og hashet som ble lagret for ham i databasen. Dette gjøres etter at vi har verifisert at eposten/personnummeret eksisterer. Vi splitter strengen på det unike tegnet slik at vi har saltet og hashet hver for seg. Så gjennomfører vi prosedyren over med det brukerinnskrevne passordet: legger til saltet og hasher strengen. Nå kan vi sammenligne dette hashet med hashet vi hentet fra databasen. Siden vi brukte det opprinnelige saltet, vil de to hashene være like dersom brukeren skrev inn riktig passord, og ulike ellers. Slik verifiserer vi at passordet er riktig på en sikker måte.

6.4 Database



Figur 21: Oppsettsfilene til databasen

Beskrivelse av filene i figur 21:

`appdb.sql`: Strukturen for applikasjonens database. Ved oppstart lager MariaDB en database med dette filnavnet (appdb).

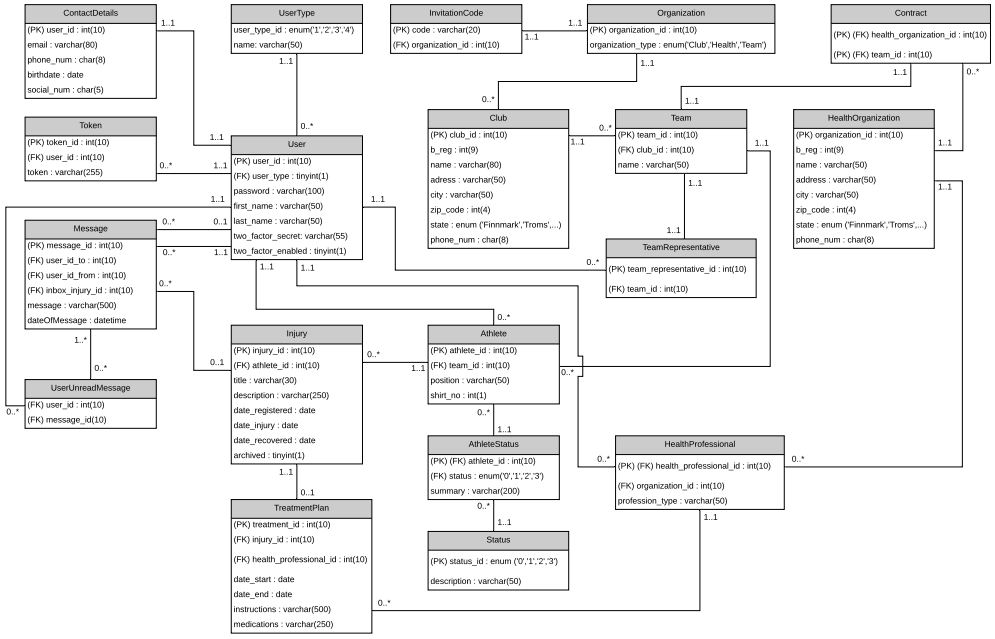
`client-setup.sql`: Setter opp databaseklienten `user` med toveis TLS-kryptering (MariaDB bruker selvsignerte OpenSSL-sertifikater).

`pma-setup.sql`: Legger inn en egen database for phpMyAdmin-funksjonalitet, siden dette ikke er inkludert i Docker imaget for phpMyAdmin.

`test-data.sql`: Testdata i applikasjonens database med brukere, organisasjoner m.m.

6.4.1 E/R-modell

Figur 22 viser forholdene til tabellene i databasen definert i `appdb.sql`. Alle kolonner som slutter på "_id" er `UNSIGNED`.



Figur 22: Databasemodell

I Tabellene `Organization` og `HealthOrganization` er kolonnen `state` lagt inn som hardkodete enumer. I systemet blir ikke dette brukt i praksis, og vi valgte derfor ikke lage en egen tabell `State` med fremmednøkler knyttet til denne.

6.4.2 Stored procedures

Systemet har følgende forhåndsdefinerte prosedyrer i databasen:

get_login_details_by_email Ved innlogging brukes denne prosdyren for å sjekke om

brukerens epost er lagret i databasen. Prosedyren returnerer `true`/ `false` og brukerens id, type, passord og 1/ 0 (tofaktorautentiseringsnøkkel er opprettet).

get_login_details_by_social_num Ved innlogging brukes denne prosdyren for å sjekke om brukerens personnummer er lagret i databasen. Prosedyren returnerer `true`/ `false` og brukerens id, type, passord og 1/ 0 (tofaktorautentiseringsnøkkel er opprettet).

create_user For registrering av en ny bruker, legger denne prosedyren inn data angående brukeren inn i tabellene `User`, `ContactDetails` og ut i fra brukertype (helsepersonell, trener eller utøver) legges den nye bruker-iden inn i tabellen `HealthProfessional`/ `TeamRepresentative`/ `Athlete`. Brukeren kobles opp mot tabellene `Organization` og `HealthOrganization`/ `Club`/ `Team` ut i fra invitasjonskoden brukeren skrev inn og type bruker.

create_club For registrering av en ny klubb legges data inn i tabellene `Organization`, `Club` og `InvitationCode`.

create_health_organization For registrering av en ny helseorganisasjon legges data inn i tabellene `Organization`, `HealthOrganization` og `InvitationCode`.

create_team For registrering av et nytt lag legges data inn i tabellene `Organization`, `Team` og `InvitationCode`.

add_new_message_athlete Ved ny melding angående en utøver legges data inn i tabellen `Message`, kolonnen `inbox_injury_id` er `NULL`. Prosedyren henter ut alle mottakere av meldingen (helsepersonell knyttet til utøveren og/eller utøveren) og legger inn den nye meldingens id inn i tabellen `UserUnreadMessage`. Prosedyren returnerer meldingens id, og fornavn og etternavn på avsender av meldingen.

add_new_message_injury Ved ny melding angående en skade legges data inn i tabellen `Message`, kolonnen `user_id_to` er `NULL`. Prosedyren henter ut alle mottakere av meldingen (skadens eier; utøveren og/eller helsepersonell knyttet til utøveren) og legger inn den nye meldingens id inn i tabellen `UserUnreadMessage`. Prosedyren returnerer meldingens id, og fornavn og etternavn på avsender av meldingen.

6.4.3 Invitasjonskoder

I kravspesifikasjonen, kapittel 3, har vi beskrevet hvordan vi vil benytte invitasjonskoder for å knytte brukere opp mot organisasjoner. Invitasjonskoder skal være unike og hemmelige, men i databasen lagres de i klartekst. De blir brukt som nøkkelen til å knytte for eksempel helsepersonell opp mot helseforetak, og hvis en hypotetisk angriper får tak i invitasjonskoden for et helseforetak, kan han lage en ny bruker der han utgir seg for

å være helsepersonell og systemet vil godta ham som det. Dette vil urettmessig gi ham mange privilegier i systemet.

Ved videreutvikling mener vi derfor at invitasjonskoder burde krypteres på samme måte som passord.

6.5 Push-varslinger

6.5.1 Firebase Cloud Messaging og pushvarsler

Vi har implementert *Firebase Cloud Messaging* (FCM) i appen med formål om å sende pushvarsler til brukerne. For å gjøre dette måtte vi lage et prosjekt i Firebase og oppgi URL-en til appen vår. Domenenavn og gyldig HTTPS er derfor forutsetninger.

Slik fungerer det: Når en bruker logger inn i appen for første gang på en ny enhet, spør appen om tillatelse til å vise varslinger. Hvis tillatelse gis, registreres app-instansen med en ID som lagres i databasen sammen med den påloggede brukeren. En bruker kan ha flere app-instanser, for eksempel hvis han bruker appen på flere enheter.

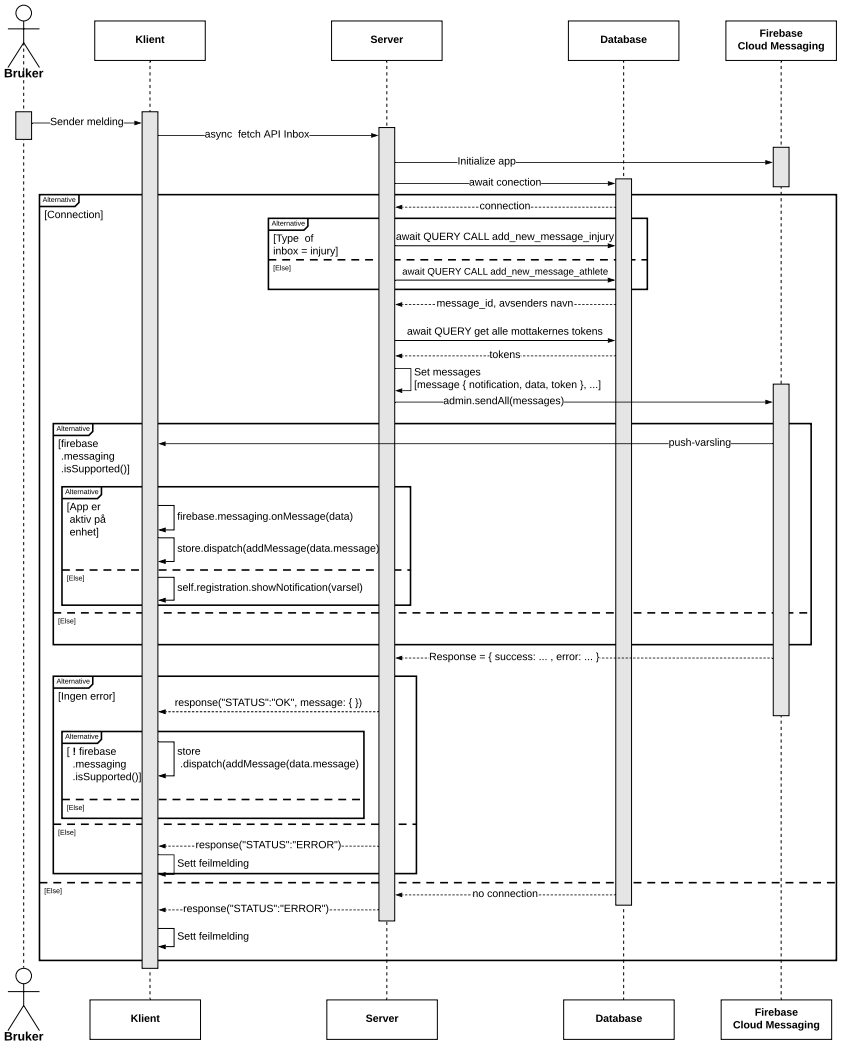
Brukeren er altså registrert med en eller flere instans-ID'er, og når vi ønsker å sende ham en pushvarsel henter vi alle disse ID-ene fra databasen og sender en FCM-melding til hver av dem. Da får han en varsel på alle enhetene der appen ikke er aktiv.

Oppførselen til pushvarsler er annerledes når appen er aktiv: vi kan motta og håndtere slike hendelser som vi vil, det blir ikke sendt en varsling automatisk. Derfor bruker vi FCM også til å sende og motta meldinger i sanntid. Når appen er aktiv, og en ny melding kommer inn til brukeren, oppdaterer vi komponentet som viser meldinger slik at den nye meldingen vises til brukeren uten at han må oppdatere siden.

FCM bruker en egen service worker til å lytte etter og håndtere bakgrunnsmeldinger, dvs. pushvarsler. Når appen er aktiv, kjøres et skript i forgrunnen som lytter etter pushvarsler.

6.6 Kommunikasjon mellom klient, server og database

For å vise hvordan kommunikasjonen mellom klient, server og database fungerer i praksis, har vi laget et sekvensdiagram, figur 23, som viser hvordan en melding sendes fra en innlogget bruker i web-grensesnittet. I eksempelet er det også naturlig å vise hvordan FCM brukes i systemet for push-varslinger.



Figur 23: Sekvensdiagram for sending av meldinger

Brukeren sender en melding fra komponenten `<inbox-content>`. Gjennom rot-reduceren `server-fetch` sendes en HTTPS-forespørsel til serverns `inbox` API. I API-en initialiseres FCM, og database-koblingen hentes.

På bakgrunn av om brukeren sendte en melding i en hovedtråd (angående utøveren) eller i en skade sendes en query som kaller på en stored procedure; enten `add_new_message_injury` eller `add_new_message_athlete`. En ny rad legges inn i tabellen `Melding`, og en rad for hver mottaker av meldingen i tabellen `UserUnreadMessage`. Stored proceduren returnerer den nye meldingens id, og fornavn og etternavn på avsenderen av meldingen.

API-en sender så en spørring for å hente token til mottakerne og avsenderen av meldingen. For hver token lages et meldings-objekt som inneholder `notification`, `data` og en `token`.

Meldingene sendes til FCM med metoden `admin.sendAll(messages, tokens)`. FCM sender ut push-varslinger med innholdet i meldingene til hver token.

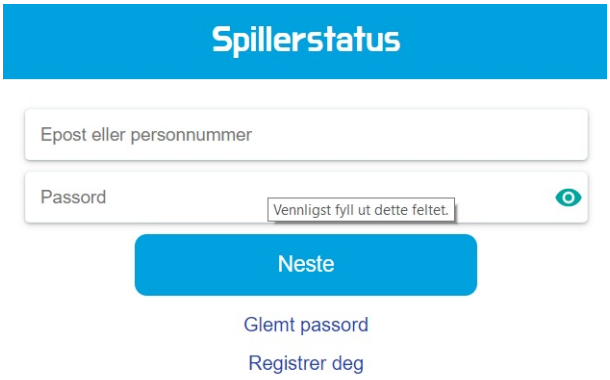
Service-workeren for FCM mottar push-varslingene. Hvis applikasjonen er aktiv på mottakerens enhet, mottar applikasjons-shellet <my-app> varselet, og dispatcher action `ADD_MESSAGE` og melding lik innholdet i `data` til rot-reduceren `inbox` slik det er beskrevet i seksjon 6.2.3. Hvis applikasjonen ikke er aktiv vises innholdet i `notification` på skjermen til mottakeren.

Etter FCM har sendt alle push-varslene, sender den en `response` tilbake til API-en. API-en sender en `response = status, message` tilbake til komponenten `<inbox-content>`. For nettlesere som ikke støtter PUSH-API, dispatcher komponenten `<inbox-content>` action `ADD_MESSAGE` og melding lik innholdet i `response.message` til rot-reduceren `inbox`.

Ved en error i databasetilkoblingen sender API-en `response` med status lik ERROR".

7 Brukergrensesnitt

Siden vi ble enige tidlig med oppdragsgiver at funksjonaliteter til applikasjonen skulle prioriteres 1.7, har vi holdt oss til et enkelt design. Vi har designet det meste selv av knapper og inntastingsfelt, se figur 24. Etterhvert som vi oppdaget webcomponents.org med sitt bibliotek av webkomponenter, har vi benyttet oss noe av dette. Derfor har noen av sidene i applikasjonen noe ulike design.



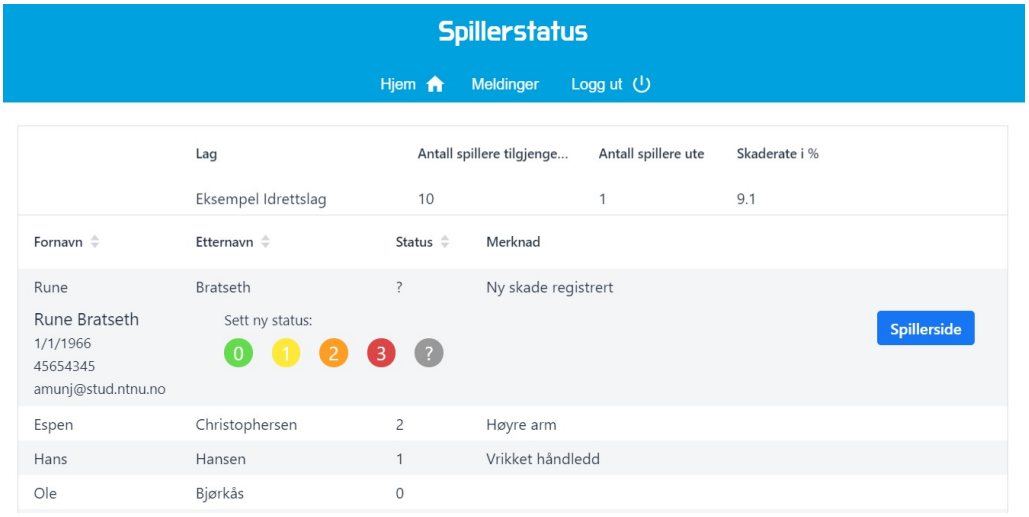
Figur 24: Innlogging. Knapp og inntastingsfelt med passordøye, eget design

7.1 Åpen kildekode webkomponenter

Via webcomponents.org har vi valgt å bruke webkomponenter fra Polymer [60] og Vaadin [61]. Vi valgte Vaadin på grunnlag av antall nedlastninger via webcomponents.org, god dokumentasjon og enkel implementasjon. Dessuten synes vi de har et attraktivt design. Webkomponentet Polymer app-layout var allerede inkludert i *PWA Starter Kit*, mer om dette i 7.1.1. Som tidligere beskrevet i seksjon 4.3 om webkomponenter, kan disse gjenbrukes for rask og effektiv implementering. Disse installeres med NPM og importeres i de dokumentene de brukes med `import`, for eksempel `import '@vaadin/vaadin-button';`. Målgruppen er som nevnt i seksjon 1.5 utøvere, trenere og helsepersonell, og de vil ha tilgang til forskjellig innhold.

I figur 25 har vi brukt `vaadin-grid` [62], en tabell som inneholder oversikt over alle utøvere på et lag, og `vaadin-button` for 'Spillerside'. Mer forklaring om hvordan sidene er bygd opp for de ulike målgruppene er i 7.2.2, 7.2.3, 7.2.4.

Dette webkomponentet kommer med ferdiglaget design og funksjonaliter. Data tilføres tabellen ved å sette tabell-elementets attributt `items`. Dette skal være en array med ob-



Figur 25: Vaadin Grid og Vaadin Button

jekter, der hvert objekt representerer en rad i tabellen. Objektene inneholder nøkkel-verdipar som representerer kolonner, der nøkkelen angir variabelnavnet for kolonnen og verdien er det som skal vises i cellen.

Vaadin Grid implementeres enkelt med `<vaadin-grid>` som vist i figur 26. Vaadin Grid har en rekke funksjonaliteter, og vi har benyttet oss av sortering av kolonner og visning av mer detaljer om hver enkelt rad. `<vaadin-grid-column>` brukes for å lage kolonner, der `path`-attributten er variabelnavnet til kolonnen i tabellens `items`.

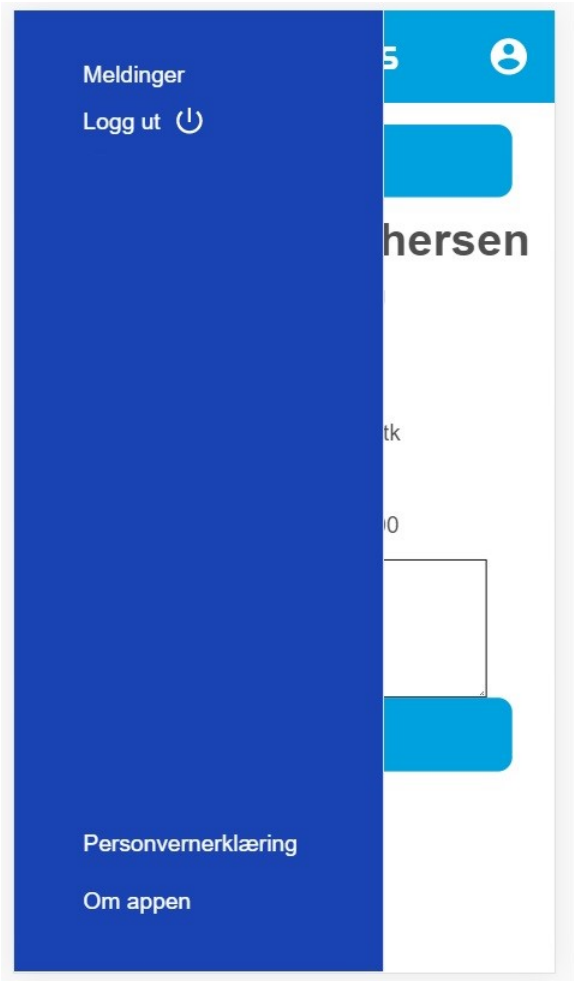
```
<div id="summary-grid-container">
  <div class="summary-grid-overlay" @click="${this._toggleTable}"></div>
  <vaadin-grid id="summary-grid" theme="no-row-borders" height-by-rows>
    <vaadin-grid-column path="" width="170px" flex-grow="0"></vaadin-grid-column>
    <vaadin-grid-column path="team_name" header="Lag"></vaadin-grid-column>
    <vaadin-grid-column path="num_active" header="Antall spillere tilgjengelig" width="190px" flex-grow="0"></vaadin-grid-column>
    <vaadin-grid-column path="num_injured" header="Antall spillere ute" width="145px" flex-grow="0"></vaadin-grid-column>
    <vaadin-grid-column path="injury_rate" header="Skaderate i %" width="130px" flex-grow="0"></vaadin-grid-column>
    <vaadin-grid-column path="" width="170px" flex-grow="0"></vaadin-grid-column>
  </vaadin-grid>
</div>
<div id="main-grid-container" style="display:block;">
  <vaadin-grid id="main-grid" theme="row-stripes no-row-borders" height-by-rows>
    <vaadin-grid-sort-column path="first_name" header="Fornavn" width="170px" flex-grow="0"></vaadin-grid-sort-column>
    <vaadin-grid-sort-column path="last_name" header="Etternavn" width="210px" flex-grow="0"></vaadin-grid-sort-column>
    <vaadin-grid-sort-column path="status" header="Status" width="95px" flex-grow="0"></vaadin-grid-sort-column>
    <vaadin-grid-column path="summary" header="Merknad"></vaadin-grid-column>
  </vaadin-grid>
</div>
```

Figur 26: HTML-kode for Vaadin Grid

7.1.1 Layout

Polymer app-layout er en webkomponent som er en samling av mange elementer for enkelt å strukturere applikasjoners layout. Denne er brukt blant annet av Youtube Web og webcomponents.org [63]. I vår applikasjon har vi benyttet oss av `<app-drawer>`, figur 27, `<app-header>` og `<app-toolbar>`, figur 28 og `app-scroll-effects`, figur 29, som

aktiveres i `<app-header>` sin attributt `effects="waterfall"`.



Figur 27: Polymer app-layout, app-drawer

`<app-drawer>` blir aktivert når skjermbredde er mindre enn 480 piksler, for eksempel ved bruk av mobil. Dette gjøres ved hjelp av `installMediaQueryWatcher('(min-width: 480px)');`, som lytter til endringer i skjermbredde og responderer når skjermbredden passerer 480 piksler. `<app-drawer>` erstatter da `<app-toolbar>`. Denne er satt for enkelhetsskyld til 480px fordi denne bredden innbefatter de fleste dagens mobiler.

Figur 28: Polymer app-layout, app-header og app-toolbar

Elementet `<app-header>` ligger på toppen av siden og innkapsler `<app-toolbar>`, og kan blant annet ha rulle-effekter. Vi har valgt kun å ha en enkel overskrift her, der overskriften

er for den aktuelle siden. I figur 29 ser man effekten av `effects="waterfall"` når man ruller nedover på siden. Elementet `<app-toolbar>` er en horisontal verktøylinje som kan innheholde, i vårt tilfelle ikoner, elementer som kan brukes til navigasjon, søkebar, handlinger, etc.

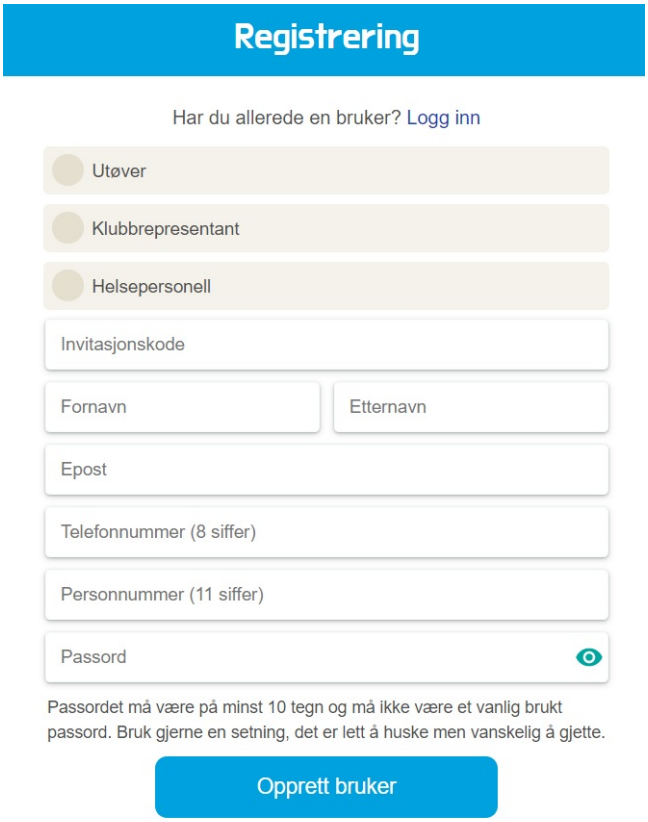
Figur 29: Polymer app-layout, scroll-effect: waterfall

7.2 De ulike brukergrensesnittene

Som nevt i 7.1 har de forskjellige målgruppene tilgang til forskjellig innhold, men de har også noe felles.

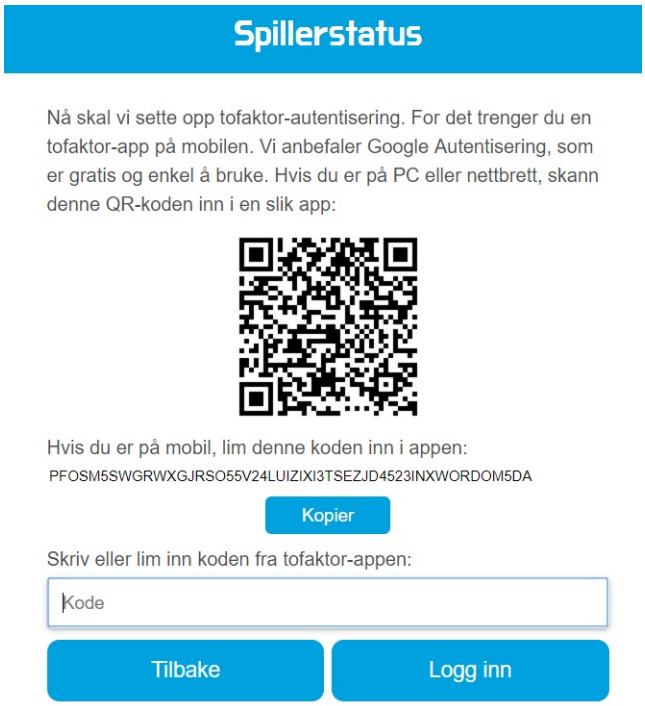
7.2.1 Felles brukergrenesnitt

Siden for innlogging er den første siden brukeren møter, figur 24. Hvis man ikke allerede er registrert, må man registrere seg ved å oppgi invitasjonskoden, se figur 30



Figur 30: Skjema for å opprette ny bruker

E-post, telefonnummer og personnumer er satt som unike verdier i databasen og man må lage seg et passord, 6.3.4, og det er sjekk om gydldig inntastet data. Oppsett av tofaktor-autentisering, 6.3.3, er neste steget i registreringen, figur 31. Etter innlogging kommer brukerene inn på sine respektive førstesider, som da vi være forskjellig ut i fra hvilken type bruker man er.



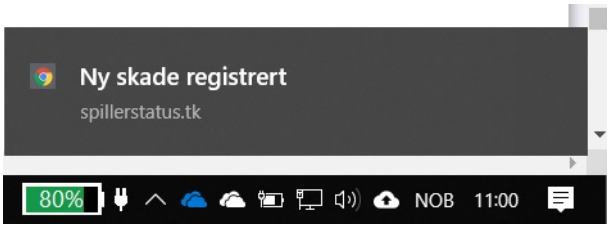
Figur 31: Side for oppsett av tofaktorautentisering

7.2.2 Utøvere

Utøvere har tilgang til et helt enkelt brukergrensesnitt, figur 32, der man har tre funksjonaliteter; registrere en ny skade, figur 34, se tidligere skader man har registrert, 35, og en side for å sende og lese meldinger, 33

Idrettslaget/klubben til utøveren er knyttet opp mot invitasjonskoden man registrerte seg meg. Utøverens status vil forandre seg fortløpende etter som et helsepersonell behandler skaden. I dette tilfellet har utøvren registrert en ny skade og er symbolisert med en grå sirkel. Det vil si at helsepersonellet enten ikke sett meldigen om den nye skaden eller ikke tatt en vurdering enda. Helsepersonell som er knyttet opp mot utøven vil få en melding straks skaden er registrert, figur 33. Melding kommer som en pop-up, enten på mobil eller pc. Det avhenger hvor man har logget seg inn tidligere. Har man logget seg inn på både pc og mobil, får man beskjed begge steder. Hvem meldingen er i fra vises ikke, og man må logge seg inn for å lese meldingen.

Figur 32: Forside for utøver



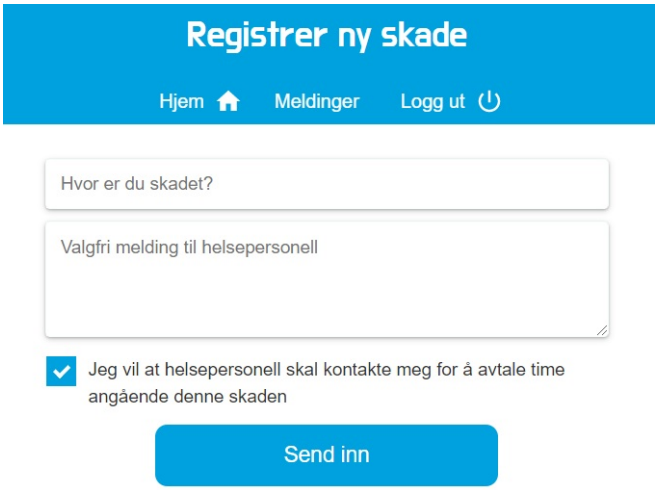
Figur 33: Meldingsbeskjed

Skaderegistrering

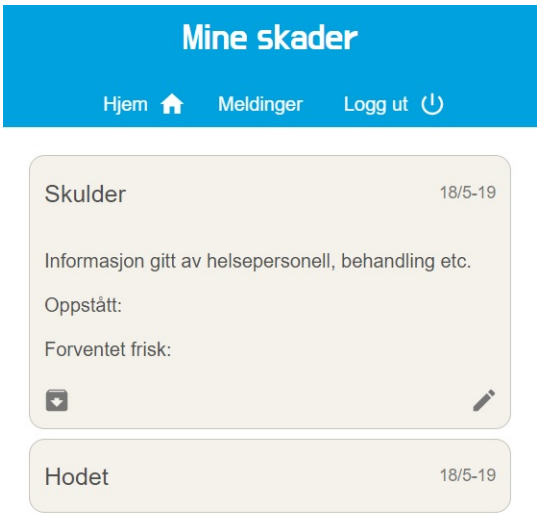
Ved registrering av ny skade, figur 34, skriver brukeren selv inn hvor skaden er og kan eventuelt legge til en kommentar. Det kan også hukes av for om man vil bli kontaktet for å avtale time i forbindelse med denne skaden. Hvis ikke, blir skaden registrert og vurdert som en smertetilstand som det ikke er behov for nærmere undersøkelse i første omgang.

Skadeoversikt

Her har brukeren en oversikt over hvilke skadre man har registrert, figur 35, dato for registrert skade og informasjon gitt av helsepersonell. Man har mulighet til å forandre på navnet på skaden hvis man for eksempel har skrevet feil og man kan velge å arkivere en skade som er ferdig behandlet. Disse skadene vil da legge seg nederst og man får en bedre oversikt etter som nye skader blir registrert i framtiden.



Figur 34: Registrering av ny skade



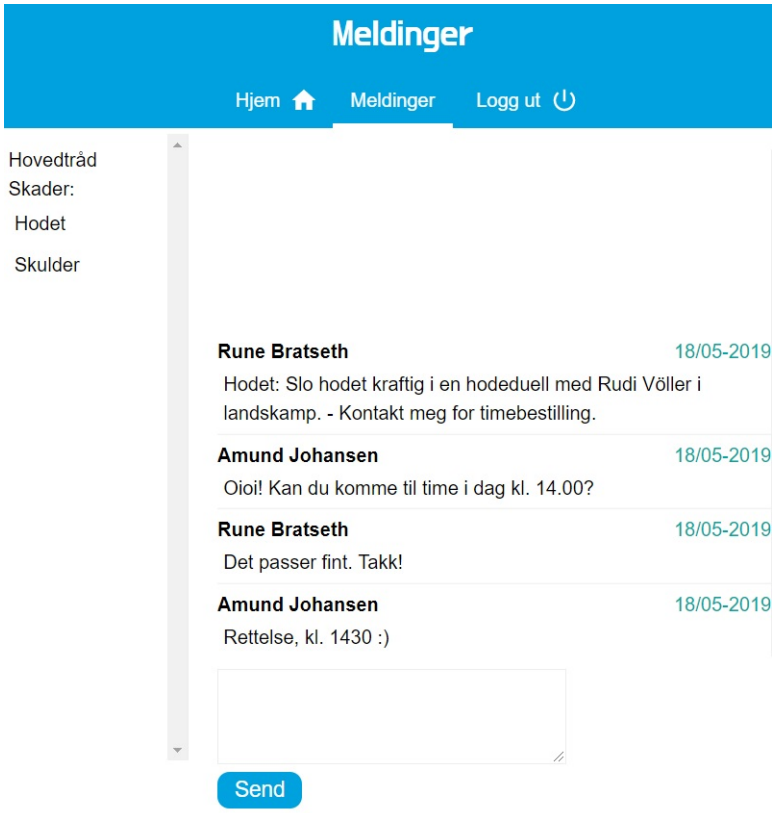
Figur 35: Oversikt over registrerte skader

Meldingsside

For hver skade som blir registrert opprettes det automatisk en meldingstråd der utøver og helsepersonell kan kommunisere. Den første meldingen baserer seg på informasjon gitt ved registrering av nye skade. Figur 36. Hvis man har mottat melding vises dette i tollbaren, se figur 39.

7.2.3 Trenere

Trenere har tilgang til en enkel oversikt over spillegruppen sin med utfyllende personlig informasjon og status samt statistikk. Dette er en litt forenklet utgave av hva helsepersonell har, 7.2.4.



Figur 36: Oversikt over meldinger per skade og sende ny melding

7.2.4 Helsepersonell

Etter innlogging får helsepersonell øyeblikkelig oversikt over de lagene de har avtale med, figur 38. Hvis det er flere lag per klubb, blir disse altså delt inn i lag.

Lagene kan kolapses og åpnes ved å trykkes på, se figur 25 og figur 38. Som nevt i 7.1 kan noen av kollonnene sorteres; fornavn, etternavn og status, og en mer detaljert visning hvis man trykker på en utøver, og tilgang til spillersiden. Her kan man raskt forandre status på en skade ved å trykke på en av de fargede sirklene. Det er kun helseperosnell som kan forandres en spillers status. Hvis det har kommet nye meldinger vises dette i `tool-baren`, se figur 39.

Vi har valgt å dele inn skjemaet i to deler; overskrift med grunnleggende informasjon om hvert lagt og statistikk, og en hoveddel, oversikt over alle spillere.

Spillerside

Spillersiden er delt inn i to deler. En side for visning av all personlig informasjon, 40, og en side for raskt å kunne sende melding om oppsatt time, figur 41

Her kan man velge å skrive meldingen manuelt, eller å bruke en dato-velger og tidspunkt-

Figur 37: Forside for trener

Figur 38: Forside for helsepersonell

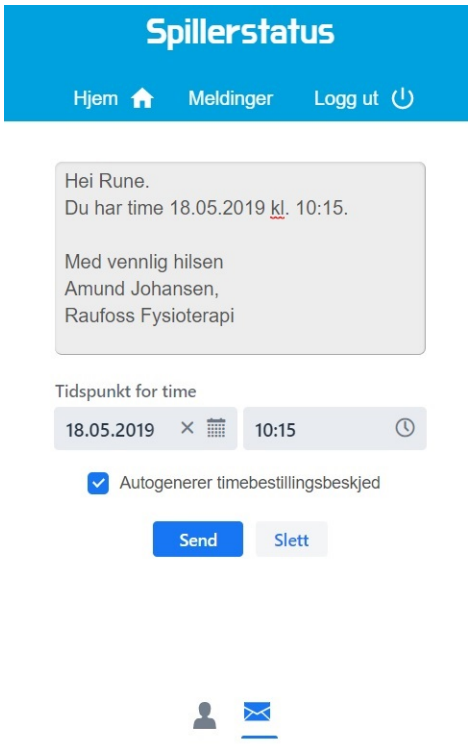
velger samt huke av for å generere meldingen automatisk.

Her kan man velge å skrive meldingen manuelt, eller å bruke en dato-velger og tidspunkt-velger samt huke av for å generere meldingen automatisk. Sistnevte kan kun genereres hvis både dato og tidspunkt er valgt, 42.

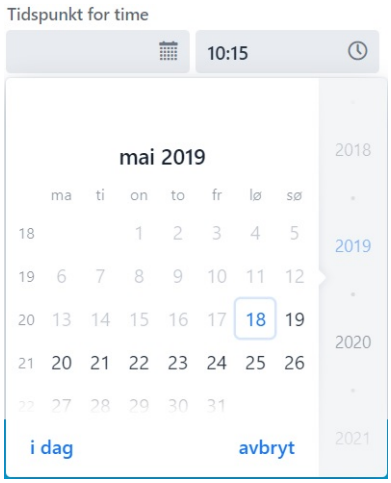
Når man trykker 'Send', vises en dialog, Vaadin dialog, for å sikre at man ikke sender for ekempel en uferdig meldig ved et uhell, figur 43. Hvis man trykker 'Ok' kommer en bekreftelse på at meldingen er sendt, figur 44.

Figur 39: Antall nye meldinger vises i toolbaren

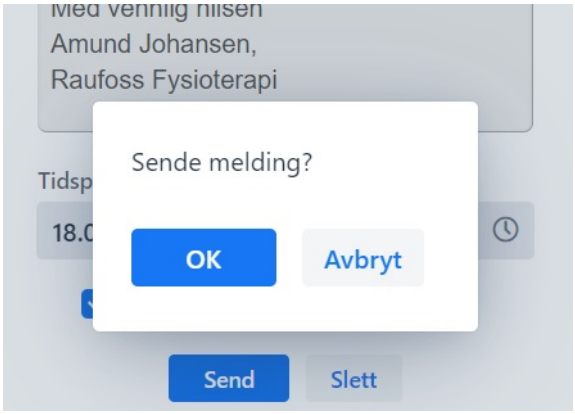
Figur 40: Spillerside med informasjon om utøveren



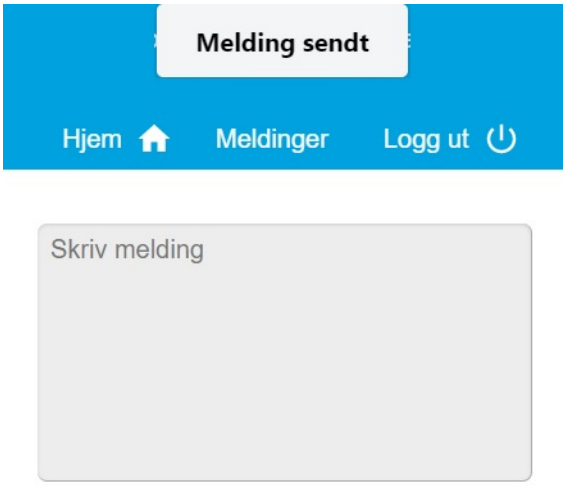
Figur 41: Spillerside med mulighet for autogenerering av melding



Figur 42: Vaadin Date Picker, tilpasset til norsk



Figur 43: Vaadin Dialog



Figur 44: Vaadin Confirm Dialog

8 Konklusjon

8.1 Resultat

Gjøvik Idrettsmedisinske team v/Terje Løkken ønsket å utvikle en prototype av en applikasjon for å holde oversikt over skader samt sikker kommunikasjon mellom helsepersonell, trenere og utøvere. Dette har vi oppnådd i en enkel form.

Følgende grunnleggende funksjoner er på plass, se seksjon 3.1:

- Alle: registrere seg selv, oversikt over skader/egne skader, sikker innlogging
- Trener: lagoversikt med spillerstatus
- Utøver: Registrere skade, meldingsutveksling, meldingsutveksling knyttet til en skade
- Helsepersonell: oversikt klienter, meldingsutveksling, meldingsutveksling knyttet til en skade, endre spillerstatus og informasjon om klient

Vi vurderer det til at vi leverer fra oss en god grunnplattform som det er fullt mulig å utvikle videre fra om ønskelig.

8.1.1 Videre arbeid

Siden dett er er en pilotversjon vil det kreve en god del mer arbeid for å gjøre applikasjonen produksjonsklar. Dette krever økonomisk kapital og vilje om videre utvikling fra oppdragsgiver.

Funksjonaliteter som ikke ble innfridd:

- For trener: opprette utøver og endre brukeropplysninger
- For utøver: endre brukeropplysninger
- For helsepersonell: endre brukeropplysninger, registrere skade, opprette klubb/lag, opprette helsepersonell/utøver, administrere skade og administrere behandlingsplan.

Disse funksjonalitetene ble lagt til underveis i prosjektet i samarbeid med oppdragsgiver, og er ikke sentrale for den grunnleggende tanken bak applikasjonen.

Videre vurderer vi disse punktene som viktige for videre arbeid med applikasjonen:

- Implementere integrasjons- og unittester
- Push-varsling for Safari i iOS

- Statisikk- og analyseverktøy:
  - i forhold til skader, rehabiliteringstid etc.
  - i forhold til bruken av applikasjonen. Hvor ofte blir den brukt?
- BankID som tofaktorautentisering

8.2 Diskusjon

Samlet sett er vi fornøyd med prototypen vi har utviklet, og rapporten vi har skrevet. Vi hadde nesten ingen virkelige kunnskaper om webutvikling da vi startet men sitter igjen med mye kunnskap – læringskurven har vært bratt. Basiskunnskaper og noen spesielle kunnskaper vi har tillært oss gjennom studiet har vi dratt god nytte av. Samtlige på gruppen synes dette har vært en interessant og lærerik prosess, selv om det til tider har vært tyngre perioder med litt frustrasjon. Vi føler vi sitter igjen med mye kunnskap, både på det tekniske plan men også i forhold til planlegging, utførelse og samarbeid med en reell kunde. I rapporten har vi nevnt hvordan noen deler kunne blitt gjort annerledes. Det er likevel noen punkter vi vil utdype:

Vi har samarbeidet godt under prosjektet, og har sittet mye sammen under utviklingen av prototypen. Trello-tavlen ble lite brukt som et resultat av god kommunikasjon i gruppen. Dette ga oss litt dårligere oversikt over det store bildet for progresjonen til prototypen, og prosjektet generelt.

Spørreundersøkelsen vi valgte å utføre tidlig i prosjeket fikk ikke det resultatet vi hadde håpet på med få svar. Likevel har diskusjonene med oppdragsgiver gitt noen synspunkter til funksjonaliteten som kanskje ikke hadde kommet frem uten spørreundersøkelsen.

Den største mangelen i prosjektet er skikkelig testing. Vi startet for sent med å finne ut hvordan dette skulle gjennomføres med de teknologiene vi valgte, og skjønte da at dette ikke var noe vi fikk tid til. Dessuten var vi veldig fokuserte på utviklingen av funksjonaliteter fra uke til uke, med demonstrasjoner på møtene med oppdragsgiver.

Bibliografi

[1] 06 2018. Veileder for vurdering av personvernkonsekvenser [internett]. Tilgjengelig fra: https://www.datatilsynet.no/regelverk-og-verktoy/veiledere/vurdering-av-personvernkonsekvenser. [Hentet jan 2019].

[2] 2019. Triage - wikipedia [internett]. Tilgjengelig fra: https://no.wikipedia.org/wiki/Triage. [Hentet 8. feb. 2019].

[3] 06 2018. Personopplysningsloven [internett]. Tilgjengelig fra: https://lovdata.no/dokument/NL/lov/2018-06-15-38. [Hentet jan 2019].

[4] Hva er en personopplysning? [internett]. Tilgjengelig fra: https://www.datatilsynet.no/rettigheter-og-plikter/personopplysninger/. [Hentet jan 2019].

[5] 01 2019. W3counter: Global web stat [internett]. Tilgjengelig fra: https://www.w3counter.com/globalstats.php. [Hentet jan 2019].

[6] 01 2019. Javascript versions [internett]. Tilgjengelig fra: https://www.w3schools.com/js/js_versions.asp. [Hentet jan 2019].

[7] 2019. Produkter - kit management [internett]. Tilgjengelig fra: http://http://www.kit-manager.no/produkter/. [Hentet 8. feb. 2019].

[8] 2019. Sideline sports/xps network [internett]. Tilgjengelig fra: https://www.sidelinesports.com/nb/. [Hentet 8. feb 2019].

[9] Børsting, J. 2017. Metoder for datainnsamling: Spørreundersøkelser, intervju og fokusgrupper [internett]. Tilgjengelig fra: https://www.uio.no/studier/emner/matnat/ifi/INF2260/h17/timeplan/chapter_5_8-norsk.pdf. [Hentet 13. februar 2019].

[10] 2000. Sjekkliste for spørreskjema [internett]. Tilgjengelig fra: http://www.xn--sprreunderskelser-10bj.no/sjekkliste-sporreskjema/. [Hentet 13. februar 2019].

[11] Bergsager, D., Brodwall, J., Hegna, A., Klever, K., Nortug, R., Saltkjel, E., & Storvik, E. 06 2018. Veileder for programvareutvikling med innebygd personvern [internett]. Tilgjengelig fra: https://www.datatilsynet.no/regelverk-og-verktoy/veiledere/programvareutvikling-med-innebygd-personvern. [Hentet jan

2019].

[12] MDN Web Docs. Push api [internett]. Tilgjengelig fra: https://developer.mozilla.org/en-US/docs/Web/API/Push_API. [Oppdatert 10. mai 2019, hentet 13. mai 2019].

[13] About notifications for websites [internett]. Tilgjengelig fra: https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/NotificationProgrammingGuideForWebsites/Introduction/Introduction.html#//apple_ref/doc/uid/TP40013225-CH1-SW1. [Oppdatert 4. september 2018, hentet 15. februar 2019].

[14] Jscrambler. April 2017. 12 frameworks for mobile hybrid apps [internett]. Tilgjengelig fra: https://blog.jscrambler.com/10-frameworks-for-mobile-hybrid-apps/. [Hentet 11. februar 2019].

[15] Firtman, M. Google play store now open for progressive web apps [internett]. Tilgjengelig fra: https://medium.com/@firt/google-play-store-now-open-for-progressive-web-apps-ec6f3c6ff3cc. [31. januar 2019, hentet 11. februar 2019].

[16] Make progressive web app (pwa) your best friend - yudiz solutions - medium [internett]. Tilgjengelig fra https://medium.com/yudiz-solutions/make-progressive-web-app-pwa-your-best-friend-3da4b4455ff7. [Hentet 19. mai 2019].

[17] Pwa stats [internett]. Tilgjengelig fra https://www.pwastats.com/. [Hentet 19. mai 2019].

[18] Progressive web applications- wikipedia [internett]. Tilgjengelig fra https://en.wikipedia.org/wiki/Progressive_web_applications. [Hentet 19. mai 2019].

[19] Google developers: Progressive web apps [internett]. Tilgjengelig fra: https://developers.google.com/web/progressive-web-apps/. [Hentet 11. februar 2019].

[20] The polymer project [internett]. Tilgjengelig fra https://www.polymer-project.org/. [Hentet 16. mai 2019].

[21] The Polymer Project. 2018. Pwa starter kit - starter templates for building full-featured progressive web apps from web components [internett]. Tilgjengelig fra: https://pwa-starter-kit.polymer-project.org/. [Hentet 22. februar 2019].

[22] Pwa starter kit [internett]. Tilgjengelig fra https://pwa-starter-kit.

polymer-project.org/. [Hentet 19. mai 2019].

[23] Litelement [internett]. Tilgjengelig fra: https://lit-element.polymer-project.org/guide. [Hentet 18. mai 2019].

[24] Templates - litelement [internett]. Tilgjengelig fra: https://lit-element.polymer-project.org/guide/properties. [Hentet 18. mai 2019].

[25] Templates - litelement [internett]. Tilgjengelig fra: https://lit-element.polymer-project.org/guide/templates. [Hentet 18. mai 2019].

[26] Introduction - lit-html [internett]. Tilgjengelig fra: https://lit-html.polymer-project.org/guide. [Hentet 18. mai 2019].

[27] Web components | mdn [internett]. Tilgjengelig fra https://developer.mozilla.org/en-US/docs/Web/Web_Components. [Hentet 19. mai 2019].

[28] Lifecycle - litelement [internett]. Tilgjengelig fra: https://lit-element.polymer-project.org/guide/lifecycle#firstupdated. [Hentet 18. mai 2019].

[29] Web components|mdn [internett]. Tilgjengelig fra https://developer.mozilla.org/en-US/docs/Web/Web_Components. [Hentet 13. mai 2019].

[30] Html imports - web components|mdn [internett]. Tilgjengelig fra https://developer.mozilla.org/en-US/docs/Web/Web_Components/HTML_Imports. [Hentet 14. mai 2019].

[31] Fronteers 2011 - fronteers [internett]. Tilgjengelig fra: https://fronteers.nl/congres/2011. [Hentet 14. mai 2019].

[32] About me - infrequently noted [internett. Tilgjengelig fra https://infrequently.org/about-me/. [Hentet 14. mai 2019].

[33] Web components - wikipedia [internett]. Tilgjengelig fra https://en.wikipedia.org/wiki/Web_Components. [Oppdatert 12. april 2019, hentet 14. mai 2019].

[34] Using custom elements - web components|mdn [internett]. Tilgjengelig fra https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_custom_elements. [Hentet 13. mai 2019].

[35] Using shadow dom - web cpmponents|mdn [internett]. Tilgjengelig fra https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_shadow_DOM. [Hentet 13. mai 2019].

[36] Using templates and slots - web components|mdn [internett]. Tilgjengelig fra https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_templates_and_slots. [Hentet 13. mai 2019].

[37] Osmani, A. The prpl pattern [internett]. Tilgjengelig fra: https://developers.google.com/web/fundamentals/performance/prpl-pattern/. [Hentet 15. februar 2019].

[38] Grigorik, I. & Surma. Introduction to http/2 [internett]. Tilgjengelig fra https://developers.google.com/web/fundamentals/performance/http2/. [Hentet 15. februar 2019].

[39] Sonarqube [internett]. Tilgjengelig fra https://www.sonarqube.org/. [Hentet 19. mai 2019].

[40] Nodejs [internett]. Tilgjengelig fra https://nodejs.org/en/. [Hentet 12. februar 2019].

[41] Github - polymer/prpl-server: An http server for node designed to serve prpl apps in production. [internett]. Tilgjengelig fra https://github.com/Polymer/prpl-server. [Hentet 29. mars 2019].

[42] Express - node.js web application framework [internett]. Tilgjengelig fra https://expressjs.com/. [Hentet 29. mars 2019].

[43] w3schools introduction to sql [internett]. Tilgjengelig fra: https://www.w3schools.com/sql/sql_intro.asp. [Hentet 6. mai 2019].

[44] Mariadb [internett]. Tilgjengelig fra: https://en.wikipedia.org/wiki/MariaDB. [Oppdatert 2. mai 2019, hentet 6. mai 2019].

[45] phpmyadmin [internett]. Tilgjengelig fra https://en.wikipedia.org/wiki/PhpMyAdmin. [Hentet 16. mai 2019].

[46] Swan, B. 2 2011. Do stored procedures protect against sql injection? [internett]. Tilgjengelig fra: https://blogs.msdn.microsoft.com/brian_swan/2011/02/16/do-stored-procedures-protect-against-sql-injection/. [Hentet 7. mai 2019].

[47] Google Developers. Firebase helps mobile and web app teams succeed [internett]. Tilgjengelig fra: https://firebase.google.com/. [Hentet 14. mai 2019].

[48] Google Developers. Build a progressive web app with firebase, polymerfire and polymer components [internett]. Tilgjengelig fra https://codelabs.developers.google.com/codelabs/polymer-firebase-pwa/. [2018, hentet 14. mai 2019].

[49] Google Developers. Cloud messaging [internett]. Tilgjengelig fra: https://firebase.google.com/products/cloud-messaging/. [Hentet 14. mai 2019].

[50] speakeasy - npm [internett]. Tilgjengelig fra https://www.npmjs.com/package/

speakeasy. [Hentet 18. mai 2019].

[51] Scrum guide | scrum guides [internett]. Tilgjengelig fra: https://www.scrumguides.org/scrum-guide.html. 2018 [hentet 18. januar 2019].

[52] Radigan, D. 01 2019. Kanban - a brief introduction [internett]. Tilgjengelig fra: https://www.atlassian.com/agile/kanban. [Hentet jan 2019].

[53] Straughan, G. From scrum to scrumban in 6 steps + free cheat sheet [internett]. Tilgjengelig fra: https://www.youtube.com/watch?v=fgT4AaKcBUA. 2018 [hentet 18. januar 2019].

[54] Sourcemaking. 2019. Singleton design pattern [internett]. Tilgjengelig fra: https://sourcemaking.com/design_patterns/singleton. [Hentet 30. april 2019].

[55] Sourcemaking. 2019. Observer design pattern [internett]. Tilgjengelig fra: https://sourcemaking.com/design_patterns/observer. [Hentet 30. april 2019].

[56] Dan Abramov and the Redux documentation authors. Normalizing state shape [internett]. Tilgjengelig fra https://redux.js.org/recipes/structuring-reducers/normalizing-state-shape. 2018 [Hentet 18. mai 2019].

[57] Wisniewski, C. 8 2016. Nist's new password rules: what you need to know - naked security [internett]. Tilgjengelig fra https://nakedsecurity.sophos.com/2016/08/18/nists-new-password-rules-what-you-need-to-know/. [Hentet 19. mai 2019].

[58] zxcvbn - npm [internett]. Tilgjengelig fra https://www.npmjs.com/package/zxcvbn. [Hentet 18. mai 2019].

[59] Crypto | node.js v12.2.0 documentation [internett]. Tilgjengelig fra https://nodejs.org/api/crypto.html. [Hentet 18. mai 2019].

[60] webcomponents.ord [internett]. Tilgjengelig fra https://www.webcomponents.org/collection/Polymer/elements. [Hentet 14. mai 2019].

[61] webcomponents.ord [internett]. Tilgjengelig fra https://www.webcomponents.org/search/vaadin. [Hentet 14. mai 2019].

[62] Grid|components|vaadin [internett]. Tilgjengelig fra https://vaadin.com/components/vaadin-grid. [Hentet 16. mai 2019].

[63] Polymerelements/app.layout - webcomponents.org [internett]. Tilgjengelig fra https://www.webcomponents.org/element/@polymer/app-layout. [Hentet 16. mai 2019].

NTNU

Norges teknisk-naturvitenskapelige universitet

Vår dato Vår referanse

A Prosjektavtale

Prosjektavtale

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

TERJE LØKKEN v/ GJØVIK BRETTSMEDISINSKE TEAM

(oppdragsgiver), og

ARVID JOHANSEN

JOHANNE BOGNØY

CECILIE FOSSUM

(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 7. 1. 2019 til 20. 5. 2019

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): _____

Oppdragsgivers kontaktperson (navn): TERJE LØKKEN

Student(er) (signatur): Johanne Bagnøy dato 11/12/18

Celine Fosserud dato 11/12-18

Anders John dato 11/12-18

_____ dato _____

Oppdragsgiver (signatur): Terje Løkken dato 11/12-2018

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 _____

B Prosjektplan

Prosjektplan bachelor

Johanne Bognøy, Cecilie Fossum, Amund Johansen

2019

Innhold

1 Mål og rammer

1.1 Bakgrunn

Gjøvik Idrettsmedisinske Team (GIT) er en tverrfaglig fagruppe innenfor kiropraktikk, medisin og fysioterapi. De jobber tett sammen for å finne gode løsninger for den enkelte idrettsutøver for å stille riktig diagnose og iverksette tiltak for at utøveren skal komme raskt tilbake til idretten. De er lokalisert på Gjøvik.

GIT tibyr blant annet akutt skadevurdering, postoperativ behandling, individuell treningsveiledning, manipulasjon og mobiliseringsteknikker med mer. Les mer på GITs hjemmeside. [1]

GIT uttrykker behov for en kommunikasjonsløsning mellom medisinsk helsepersonell, trenere og utøvere. Dette er i stor grad for å sikre personvern ettersom GITs arbeid involverer sensitiv, personlig informasjon. Løsningen skal være en felles plattform som forenkler og effektiviserer kommunikasjon og skaderapportering samt gir oversikt over utøvere og skade-/smertetilstander. For helsepersonell kan tidlig informasjon om smertetilstander danne grunnlag for skadeforebyggende tiltak, mens trenere vil kunne få en oppdatert oversikt over utøveres status i forhold til skade og om de er konkurranseklare.

1.2 Prosjektmål

Effektmål:

- Ved prosjektets slutt skal det foreligge en testversjon som er god nok til pilotprosjekt. I så tilfelle kan det være et salgbart produkt i løpet av 2020.
- 2021: Systemet er i bruk hos en eller flere tippeligaklubber i fotball og norsk topphåndball.

Resultatmål:

- Systemet skal forbedre rutiner og sikkerhet omkring taushetsbelagte opplysninger.
- Systemet skal gjennom kontroll og oversikt kunne forebygge skader samt redusere skadetid.

Læringsmål: Vi ønsker å lære oss prosessen ved utvikling av ny programvare ved å jobbe sammen i gruppe og i samarbeid med en oppdragsgiver. Oversikt over fokusområder:

- Programvarearkitektur
- Objektorientert programvaredesign
- Grunnleggende JavaScript
- HTML
- CSS
- Database
- Sikkerhet

- Følge en systemutviklingsmodell (Scrumban)
- Systematisk bruk av Git *repository*
- Versjonsnummerering
- Systematisk testing og kvalitetskontroll av kildekode og funksjonaliteter

1.3 Rammer

Juridiske rammer Systemet skal være i tråd med personopplysningsloven [2], og derav leve opp til kravet om innebygd personvern. Sistnevnte må involveres gjennom hele systemutviklingsprosessen.

Datatilsynet klassifiserer helseopplysninger som sensitive [3]. Derfor er vi pålagt å implementere tofaktor-autentisering på innlogging som gir adgang til helseopplysninger.

Teknologiske rammer Webgrensesnittet skal fungere på på de mest populære nettleserne per 31. desember 2018. [4] Det vil si nyere versjoner av Chrome, Safari, Internet Explorer, Edge, Firefox og Opera. Disse støtter HTML 5 og ECMAScript 5 (Javascript versjon 5). [5]

Økonomiske og tidsmessige rammer Programvaren skal være ferdig i følge planen i Gantt-diagrammet (figur 3). Utgifter til pilotprosjektet etter dette prosjektets slutt dekkes av GIT.

2 Omfang

2.1 Fagfelt/Problemområde

De fleste idrettslag i Norge i dag bruker alminnelige kommunikasjonsformer som tekstmelding, telefon, epost og Facebook. Det finnes noen digitale kommunikasjons- og skaderapporteringssystemer for idrett, som for eksempel norske Kit Manager [6] som brukes av landslaget i fotball og kanadiske Athlete Monitoring [7] som brukes av Strømsgodset Fotball. Men det er altså kun et fåtall idrettslag som har tatt slike systemer i bruk.

Vår oppdragsgiver i GIT opplever at kommunikasjonen med klienter og kolleger ikke fungerer godt nok. Kommunikasjonen deres foregår over nevnte kanaler, og som et resultat er viktig informasjon spredt rundt på ulike plattformer. Dette er ikke bare uoversiktlig for GITs ansatte, men gir også svakt personvern. GIT bruker ofte forkortelser i sin kommunikasjon, som for eksempel å referere til klienter med deres initialer, for å beskytte deres identiteter. Problemområdet er dermed tredelt: det er behov for et system som både gir oversikt og tilbyr sikker kommunikasjon, men viktigst er det kanskje at systemet er enkelt å bruke slik at det faktisk blir brukt.

2.2 Avgrensning

Vi skal utvikle en responsiv webapplikasjon som også er tilgjengelig på mobil, men ikke som en native applikasjon. Løsningen skal utvikles for å kunne utvides i ettertid; for eksempel skal den kunne integreres med GITs journalsystem eller pasientjournaler dersom det er behov for dette i fremtiden. Slike utvidelser er altså ikke en del av oppgaven, men vi skal legge til rette for dem.

Løsningen skal dreie seg om helse, og er ment for medisinsk helsepersonell og deres klienter. Den skal altså være et kommunikasjonsverktøy mellom helsepersonell og kolleger, utøvere eller trenere, det vil si at den ikke skal støtte kommunikasjon mellom utøvere og trenere. For trenere skal løsningen kun gi oversikt over spillerstatus, den skal ikke være et komplett verktøy for kamp-/treningsplanlegging. På samme måte skal den heller ikke tilby timebestilling og -planlegging for helsepersonell; løsningen kommer ved siden av GITs interne journalsystem for nå.

2.3 Oppgavebeskrivelse

Vår oppgave er å levere et webgrensesnitt for bruk i det daglige samarbeidet mellom GIT, idrettsutøvere og trenere. Produktet vi leverer ved prosjektets slutt skal være klart for gjennomføring av pilotprosjekt ved GIT.

Vi definerer følgende roller: Helsepersonell, Idrettsklubb, Utøver og Trener. Helsepersonell og Idrettsklubb klassifiseres som administratorroller, mens Utøver og Trener er vanlige brukerroller.

Systemet deles opp i høysikker (rød) og middels sikker (gul) sone. Rød sone defineres som alt som gir innsyn i eller administrativ tilgang til helseopplysninger; resten er gul sone.

Brukergrensesnittet vil være forskjellig for hver rolle, og applikasjonen skal ha følgende overordnede funksjonalitet for de forskjellige rollene:

Rolle	Funksjon
Alle	* Registrere ny bruker
	* Sikker innlogging
Helsepersonell	* Registrere klienter i systemet
	* Registrere skade for en utøver
	* Redigere/oppdatere en skade
	* Endre en utøvers spillerstatus
	* Oversikt over klienter
	* Se informasjon om en utøver
	* Meldingsutveksling med helsepersonell eller utøver
Utøver	* Registrere skade
	* Oversikt over nåværende skader
	* Skadehistorikk
	* Meldingsutveksling med helsepersonell
Trener	* Lagoversikt med spillerstatus
Idrettsklubb	* Registrere lag i systemet
System	* Loggføring

Tabell 1: Funksjoner per rolle

Med klient mener vi et idrettslag eller en idrettsutøver som er klient hos medisinsk helsepersonell. Sistnevnte kan for eksempel være GIT i sin helhet eller en av GITs ansatte.

[Alle] Registrere ny bruker Ved registrering må administratorer autentiseres spesielt. For alle kommer det til å brukes grunnleggende autentisering som for eksempel at man får et automatisk generert brukernavn og passord på grunnlag av klient-/laglister, og må deretter opprette eget passord før man er innlogget første gang. Mer om dette under.

[Alle] Sikker innlogging Etter førstegangsregistrering skal det være enkelt å logge seg inn og benytte seg av grunnleggende funksjonalitet. Gul sone skal ha en rask men likevel moderat sikker påloggingsform, som for eksempel pinkode, mens rød sone skal ha tofaktor-autentisering, muligens ved bruk av en løsning som BankID (på mobil). Det kan vise seg at BankID på mobil er enkelt nok til bruk i gul sone også, noe som vil gjøre skillet mellom soner unødvendig.

[Helsepersonell] Registrere klienter i systemet Under utrulling og brukerregistrering kan det være aktuelt å benytte følgende metode: Systemet tar en liste over personer som skal kunne bruke det og genererer et unikt brukernavn og passord til hver. Administratorer får

ansvar for å overgi disse opplysningene til de rette personene, og når disse logger seg i systemet for første gang må de sette et eget passord. Ved at disse listene gis av medisinsk helsepersonell vil man kunne verifisere hvilke personer som er klienter hos hvilke helsefaglige.

[Helsepersonell] Registrere skade for en utøver [Rød sone] I en skadesak er det ikke alltid utøveren som rapporterer skaden først; helsepersonell må også kunne gjøre dette. I så tilfelle kan det være flere opplysninger å rapportere, spesielt hvis helsefaglig allerede har behandlet skaden. Disse opplysningene skal være på en form som tillater at systemet kan behandle dem og bruke dem i statistikk.

[Helsepersonell] Redigere/oppdatere en skade [Rød sone] Helsepersonell må kunne endre eller legge til opplysninger på en registrert skade, ikke minst når den er rapportert av utøver. Opplysninger som kan registreres av helsepersonell kan for eksempel være hvilken dato skaden oppstod, hvilke behandlinger eller tiltak som er gjort, resultatene av disse osv. Det er også helsepersonell som skal arkivere en skade når den er bekreftet over.

[Helsepersonell] Endre en utøvers spillerstatus [Rød sone] Autoriteten til å bestemme en utøvers spillerstatus gis til helsepersonell. En status består av et tall på skalaen 0-3 samt en kort oppsummering av utøverens tilstand og anbefalte hensyn i forhold til sportsaktiviteter. Denne oppsummeringen skrives av helsepersonell og er ikke det samme som beskrivelsen under.

Beskrivelse av skalaen:

0	Grønn	Skadefri
1	Gul	Smertetilstand, men kampklar
2	Oransje	Skade/smertetilstand, kan delta på kollektiv trening men kan måtte ha tilpasning, ikke kampklar
3	Rød	Skade/smertetilstand, kan ikke trene kollektivt og ikke kampklar

[Helsepersonell] Oversikt over klienter [Gul sone] For helsepersonell skal systemet vise en oversikt over klienter med navn og spillerstatus for hver utøver.

[Helsepersonell] Se informasjon om en utøver [Rød sone] Ved å klikke på en utøver i oversikten skal helsepersonell komme til en side der man har tilgang til de samme funksjonene som utøvere under: Oversikt over nåværende skader og Skadehistorikk.

[Helsepersonell] Meldingsutveksling med helsepersonell eller utøver [Gul sone] Systemet skal tilby meldingsutveksling mellom helsepersonell og utøvere, samt mellom helsepersonell og kolleger. Formålet med dette er kommunikasjon på siden av skaderapportering.

[Utøver] Registrere skade [Gul sone] Utøvere skal raskt og enkelt kunne rapportere skader. Det skal være et begrenset antall opplysninger som de må oppgi. Denne funksjonen skal være en systematisk måte for utøvere å kontakte helsepersonell på, med grunnleggende informasjon som helsepersonell kan bruke i prioritering og planlegging. Det vil være en avmerkingsboks som skal avmerkes dersom dette er en skade som har gjenoppstått. Skaden identifiseres da av helsepersonell.

[Utøver] Oversikt over nåværende skader [Rød sone] For utøvere skal systemet kunne vise "Min side" der de kan se oversikt over registrerte skader og alle opplysninger knyttet til disse. Denne funksjonen er også en del av Oversikt over klienter for Helsepersonell.

[Utøver] Skadehistorikk [Rød sone] Registrerte skader deles opp i nåværende og historiske skader. I skadehistorikken finner man alle arkiverte skader. Denne funksjonen er også en del av Oversikt over klienter for Helsepersonell. I skadehistorikk vil utøveren også kunne gjenopprette en tidligere skade som har kommet tilbake.

[Utøver] Meldingsutveksling med helsepersonell [Gul sone] Dette er utøverens funksjon for meldingsutveksling. Utøvere kan ikke bruke denne funksjonen for å kommunisere med trenere eller andre utøvere.

[Trener] Lagoversikt med spillerstatus [Gul sone] Trenere får den same oversikten over utøvere med spillerstatus som helsepersonell, men har ikke tilgang til videre detaljer om hver enkelt utøver. Dette er den eneste spesielle funksjonen systemet tilbyr trenere, og det er ingen rød sone for trenere.

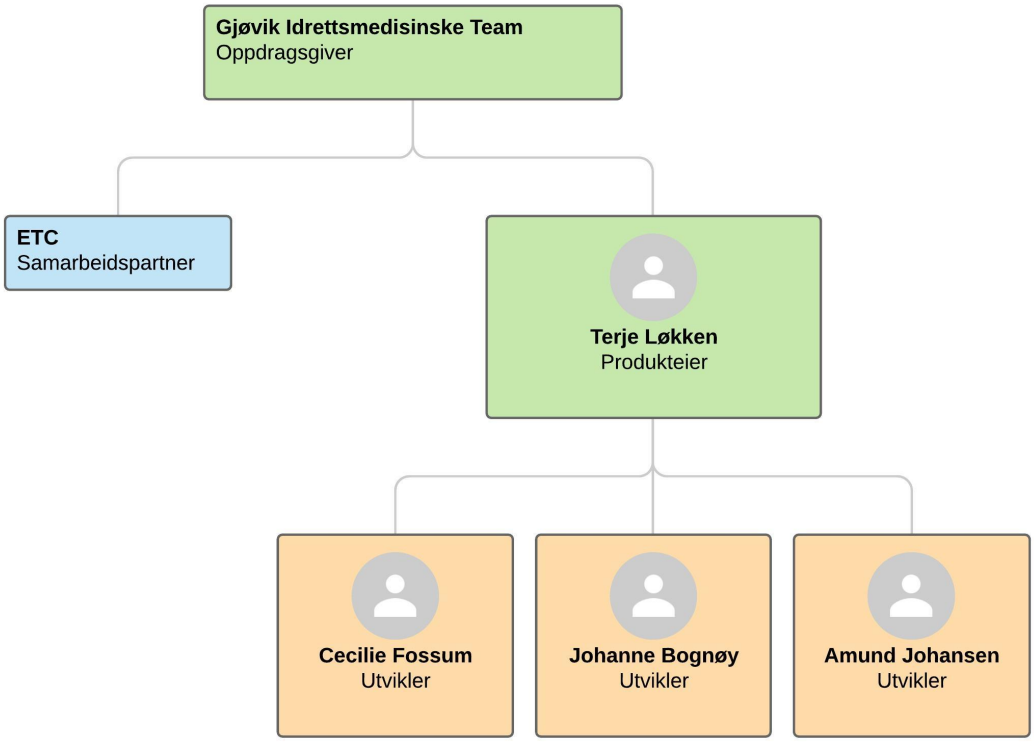
[Idrettsklubb] Registrere lag i systemet Dette vil være på samme måte som helsepersonell registrerer klienter i systemet: idrettsklubber leverer laglister med trenere og utøvere. Det vil naturligvis være overlapp mellom laglister og klientlister, og derfor må systemet kunne aggregere disse ved å matche personer. Disse aggregatene bør kontrolleres av administrator.

[System] Loggføring Systemet skal loggføre internt hvem, hva og når for all aktivitet.

Det er viktig at applikasjonen er intuitiv og enkel å bruke, slik at folk ønsker å bruke den. Derfor vil vi gjennomføre en spørreundersøkelse av GIT og deres nettverk av utøvere og trenere for å finne ut mer om hva de ser etter i en slik programvare og hva som skal til for at de skal ta den i bruk. Resultatene av denne spørreundersøkelsen vil være med på å forme oppgaven videre.

3 Prosjektorganisering

3.1 Organisasjonskart



Figur 1: Organisasjonskart

3.2 Ansvarsforhold og roller

Leder Johanne Bognøy
Kontaktperson for veileder og oppdragsgiver, hovedansvar for sikkerheten i systemet

Sekretær Cecilie Fossum
Møtereferat, hovedansvar for redigering av prosjektrapporten i LaTeX

Romansvarlig Amund Johansen
Ansvar for reservasjon av grupperom, annet praktisk

3.3 Rutiner og regler i gruppa

1. Grupperegler

(a) Alle legger inn innsats for å oppnå høy måloppnåelse

(b) Alle er forpliktet til å følge de fastlagte utviklingsrutinene og det som omfatter kvaitetssikring. Se avsnitt 5

(c) Ved planlagt fravær skal man si fra til gruppen så snart som mulig

(d) Man skal si fra til gruppen om fravær grunnet sykdom, og ved lengre sykdomsperioder skal gruppen sette inn tiltak som for eksempel omfordele arbeidsoppgavene

(e) Man skal si fra til gruppen dersom man ikke er i stand til å gjennomføre gitte arbeidsoppgaver innen satt frist

(f) Man skal si fra til gruppen dersom man står fast på en oppgave og trenger hjelp

(g) Ved faglige uenigheter skal gruppen forsøke å komme til enighet, og eventuelt fatte en flertallsavgjørelse som det kreves lojalitet til

(h) Leder er gruppens kontaktperson for veileder og oppdragsgiver

(i) Sekretær skriver referat fra hvert møte med veileder og oppdragsgiver

(j) Romansvarlig bestiller på forhånd rom til møtene hver mandag og fredag samt andre planlagte møter

2. Eksempler på alvorlige og/eller gjentatte brudd på reglene

(a) Gjentatt mer enn 30 min forsinkelse til gruppemøter fra avtalt tid

(b) Ugyldig, uanmeldt fravær til gruppemøte

(c) Alvorlig mangelfullt arbeid eller alvorlig overskridelse av fristen for en arbeidsoppgave

3. Sanksjoner/reaksjoner på regelbrudd

(a) Personen regelbruddet gjelder blir informert muntlig av et annet gruppemedlem

(b) Personen regelbruddet gjelder får en skriftlig advarsel av de to andre gruppemedlemmene

(c) Samtale mellom de to andre gruppemedlemmene og veileder

(d) Dersom problemet vedvarer, offisiell ekskludering

4 Planlegging, oppfølging og rapportering

4.1 Hovedinndeling av prosjektet

Valg av systemutviklingsmodell

Systemet vi skal utvikle har ikke fastbestemt størrelse eller funksjonalitet, og derfor trenger vi en smidig utviklingsmetode. Vi har vurdert Scrum [8] og Kanban [9] som mulige alternativer. Scrum har en god ramme som passer bra for prosjektet vårt med tanke på endringer av systemet, men bruker mye tid på estimering og faste sprinter, noe vi ikke har mye erfaring med. Kanban

Figur 2: Scrumban-tavle
[10]

er friere og åpnere, men vi finner det for åpent; vi trenger noen rammer for tidsbruk blant annet med tanke på ukentlige møter med oppdragsgiver.

Vi har dermed valgt kombinasjonen Scrumban [10] som utviklingsmodell med noen tilpasninger. Daily Scrum og Scrum planning meetings byttes ut med en hybrid vi kaller Weekly Scrum. Weekly Scrum er ukentlige møter med Product Owner, og eventuelt samarbeidspartner etter behov, hvor vi går igjennom hva vi jobber med, veien videre og eventuelle endringer.

Vi bruker en Trello-tavle (figur 2) hvor arbeidsoppgavene er skrevet på kort og kategorisert i kolonner etter progresjon. Fra Scrum har vi beholdt *product backlog* og lagt den inn på Trello-tavlen som første kolonne. Alle arbeidsoppgavene for hele prosjektet blir lagt inn i *product backlog*. Product Owner har også mulighet til å legge inn arbeidsoppgaver her. På de ukentlige møtene med Product Owner prioriteres *product backlog* med de fem viktigste oppgavene øverst.

Det er satt en grense for hvor mange kort som til enhver tid kan ligge i noen av kolonnene. Kolonnen *to do* har maks fire kort, *ready* har maks tre, *development* har maks fire, *ready to test* har maks fire og *test* har maks to. Kolonnen *to do* erstatter Sprint Backlog fra Scrum, men uten tidsestimering av oppgaver. Kort legges inn i denne kolonnen fortløpende og etter kapasitet.

Når *to do* har ledig plass trigger dette et møte med utviklergruppen hvor de øverste kortene fra *product backlog* blir flyttet over til *to do*. Videre flyttes kort fra *to do* til *ready* hvor kortene skal ligge i prioritert rekkefølge etter hva utviklergruppen kan jobbe med først.

Utviklerne velger et kort fra *ready* og jobber på det til oppgaven er klar for testing eller er

blokkert. Hvis et kort er blokkert kan utviklerne velge et nytt kort fra *ready* hvis det er plass til flere kort i *development*; hvis ikke må en eller flere av de andre utviklerne hjelpe til med å fjerne blokkeringen.

Etter vellykket testing av et kort legges det i *done*. Hvis testingen feilet, merkes kortet som *test failed* og legges i *development* uavhengig av antall ledige plasser. Utvikleren som har jobbet med kortet har ansvar for å rette opp i feilen og må prioritere dette foran andre kort.

Vi følger prinsippet *Don't test what you can't release, and don't develop what you can't test.*[10]

4.2 Plan for statusmøter

Gruppen møtes fast mandag kl. 08.30 - 15.00 og fredag kl. 8.30 - 16.00 om ikke annet er avtalt. Grupperom skal være reservert på forhånd. I startfasen av prosjektet møtes gruppen med veileder ukentlig onsdag kl. 11.00 og med oppdragsgiver torsdag kl. 9.00. Neste møte med oppdragsgiver og eventuelt samarbeidspartner avtales fortløpende.

5 Organisering av kvalitetssikring

Vi skal etterstrebe gode utviklingsrutiner under hele prosjektet.

5.1 Dokumentasjon, standardbruk og kildekode

- Rapporten skrives i LaTeX og er synkronisert med Dropbox.
- Alle andre dokumenter (møtereferat, maler etc.) lagres på Google Disk.
- Oppdragsgiver har tilgang til møtereferat via Google Disk.
- All kildekode ligger i sine respektive *branches* lokalt og/eller i Bitbucket *repository*. Se avsnitt 5.3.
- Utviklingen av programvaren (de forskjellige fasene) visualiseres i Trello. Se avsnitt 4.1.
- Valg av ulike løsninger (programvaredesign, teknologi, programmeringsspråk etc.) skal begrunnes og dokumenters i rapporten fortløpende.

Standarder

All kode skal følge de retningslinjene som er implementert i SonarQube. [11]

Testing

Det skal skrives *Unit-tester* for kritiske funksjonaliteter/komponenter av systemet. Det skal også kjøres integrasjonstester mellom ulike API-er. Etterhvert som systemet tar form vil det

være naturlig å kjøre tester underveis med ansatte ved GIT og/eller et utvalg av trenere og utøvere.

- JavaScript-kode testes i Jasmine [12]

Analyse

Kodekvaliteten analyseres med SonarQube. Dette er et verktøy for kontinuerlig analayse og støtter blant annet JavaScript. SonarQube oppdager potensielle bugs, sårbarheter, dårlig struktur, duplisering av kode, kode som er for kompleks (vanskelig å vedlikeholde), brudd på kodestandard med mer.

5.2 Utviklingsrutiner

- Alle arbeidsoppaver skal defineres og legges inn i Trello-tavlen. Se avsnitt 4.1.
- Hvert enkelt gruppemedlem har selv ansvar for at sin arbeidsoppgaves forløp blir oppdatert i Trello
- Alle *commits* i Bitbucket skal være informative
- Kommentering skal følge språkets retningslinjer
- Navnsetting av variable, klasser og funksjoner skal være informativt og følge språkets retningslinjer

5.3 Konfigurasjonsstyring

Under utviklingen av programvaren bruker vi Git som versjonkontrollsystem i Bitbucket og fem forskjellige *branches* [13] med klare regler for hvordan de skal brukes.

Hvert prosjektmedlem *puller* fra og *pusher* til *central repository* på Bitbucket, men hver enkelt kan også *pulle* fra andre i gruppen for å danne samarbeidsgrupper. Dette kan med fordel gjøres hvis man jobber med en stor funksjonalitet, i stedet for å *pushe* et pågående arbeid til Bitbucket før det er ferdig.

- Main branches
  - Master
    Utgivelsesklar kildekode. Det vil si at hver gang Master oppdateres er dette per definisjon en ny utgivelse.
  - Development
    Nye funksjonaliter under utvikling. Når kildekoden i Development er stabil og klar for utgivelse skal alle nye forandringer *merges* tilbake til Master tagget med et utgivelsesnummer.
- Supporting branches
  De forskjellige Supporting branches er til for å bistå parallell utvikling mellom prosjektmedlemmer, forenkle sporing av funksjonaliteter, forberede produksjonutgivelser og for

raskt å fikse produksjonproblemer i utvikling. I motsetning til Main branches har disse *branchene* alltid en begrenset levetid, siden de vil bli fjernet til slutt.

- Feature branches
  Essensen med en Feature branch er at den kun eksisterer så lenge funksjonaliteten er under utvikling, og vil til slutt *merges* tilbake til Development for endelig å legges til kommende utgivelse. Hvis den viser seg å ikke være tilfredstillende, forkastes den.
  - Skal kun *branches* fra Development
  - Skal *merges* tilbake til Development etter at testing er ferdig og hvis funksjonaliteten er som ønsket. Hvis ikke forkastes den.
  - Skal kun eksistere lokalt hos utvikler
  - Skal slettes fortløpende, etter at den enten er *merget* tilbake til Development eller forkastet)
  - Flagget `--no-ff` skal benyttes ved *merging* til Development. (Dette for å lage et nytt *commit*-objekt. Forhindrer å miste historisk data om en *feature branch*.)
- Release branches
  Støtter forberedelse til en ny produktutgivelse. Siste finpuss, mindre *bug fixes* og metadata (versjonnummer, utgivelsesdato) skjer her. En ny *release branch* fra Development opprettes når Development gjenspeiler ønsket tilstand for neste utgivese. Alle funksjonaliteter i *feature branches* som er planlagt for den nye utgivelsen må *merges* tilbake til Development på dette tidspunktet. Funksjonaliteter for fremtidige utgivelser skal ikke dette.
  - Skal kun *branches* fra Development
  - Skal *merges* tilbake til Development og Master
  - Små *bug fixes* kan utføres her
  - Ikke tillatt å legge til nye store funksjonaliteter. (De skal *merges* tilbake til Development og må dermed vente til neste store utgivelse.)
  - Ved starten av en ny *release branch* gis den kommende versjonen et versjonsnummer. Se avsnitt 5.4
  - Når en *release branch* er klar skal den *merges* inn i *master* og tagges med versjonsnummer. Eksempel: `--no-ff release-1.2`
  - Til slutt skal forandringer gjort i *release branchen merges* tilbake til Development slik at fremtidige utgivelser også inneholder eventuelle *bug fixes*.
  - Etter *merging* med Development vil det sannynligvis oppstå en *merge conflict*, ettersom versjonnummeret er endret. Dette skal fikses og kommenteres.
- Hotfix branches
  Veldig lik *release branches* men har sitt opphav når en kritisk feil i utgitt versjon må løses øyeblikkelig. Essensen med en sånn *branch* er at de andre i gruppen kan fortsette sitt arbeid, mens én tar seg av problemet.

* Skal kun *branches* fra Master. (Eksempel: 1.2 er nåværende versjon og forandringer i Development er ikke klare enda. Oppretter *hotfix branch* og fikser problemet. Ny versjon 1.2.1)
* Husk å lage nytt versjonsnummer ved opprettelse av ny *hotfix branch*
* Legg til kommentar til forbedringen ved *commit*
* *Merges* tilbake til Master og Development slik at utførte *bug fixes* også blir med i neste versjon.
* Unntak hvis det allerede finnes en *release branch*. Da skal den *merges* til denne og ikke Development. (Den vil uansett *merges* tilbake til Development når en *release* er ferdig).

5.4 Versjonsnummerering

Prosjektet benytter seg av Semantic Versioning v2.0.0. [14]

5.5 Risikoanalyse

#	Beskrivelse	Sannsynlighet	Konsekvens	Verdi
1	Systemet samsvarer ikke med oppdragsgivers ønsker	Liten	Middels	2
2	Sykdom i gruppen	Liten	Stor	3
3	Feilestimering av prosjektets størrelse	Liten	Stor	3
4	Bruk av for mye tid på mindre deler av prosjektet	Middels	Middels	4
5	Tap av kildekode	Liten	Kritisk	4
6	Personopplysninger lekker ut etter at systemet er tatt i bruk	Middels	Kritisk	8

Tabell 2: Risikovurderinger etter tiltak

	Liten	Middels	Stor	Kritisk
Liten sannsynlighet		1	2, 3	5
Middels sannsynlighet		4		6
Stor sannsynlighet				

Tabell 3: Verdier for risikovurdering

1. Som uerfarne utviklere, og siden dette er vårt første prosjekt for en reell arbeidsgiver, er sannsynligheten stor for at vi ikke klarer å møte oppdragsgivers ønsker for systemet.

   Vi har fra starten avtalt å ha ukentlig dialog og møter med oppdragsgiver som skal forebygge dette, men det kan lett oppstå misforståelser om produktets funksjonalitet og utforming. Gruppen skal etterstrebe god kommunikasjon og følge avtaler fra møtereferat. I tillegg skal gruppen, i samarbeid med oppdragsgiver, utføre en spørreundersøkelse for å hente inn informasjon om hva trenere, utøvere og helsepersonell ønsker seg i systemet. Hvis tidsplanen tillater det skal programvaren endres for å oppfylle oppdragsgivers ønsker; eventuelt skal man komme til enighet om andre løsninger. Sannsynligheten for å ikke møte oppdragsgivers ønsker er på bakgrunn av disse tiltakene satt til liten. Konsekvensen av å ikke møte ønskene er middels for bacheloroppgaven da det er rapporten som teller mest for oss, men ikke liten da et godt sluttprodukt gir en god rapport.

2. Sykdom innad i gruppen, og spesielt over lengre perioder, er umulig å forutse. Vi vurderer langvarig sykdom som lite sannsynlig, men om dette skulle inntreffe må man revurdere oppgavens omfang og bli enig med oppdragsgiver om dette. Ved kortvarig sykdom må kritiske arbeidsoppgaver omfordeles. Langvarig sykdom kan gi store konsekvenser for en liten utviklingsgruppe, og det er derfor viktig med god kommunikasjon i slike tilfeller innad i gruppen, med veileder og oppdragsgiver.

3. Siden vi samarbeider med en ekstern aktør, ETC, som har gitt oss gode tibakemeldinger på avgrensning av prosjektet, vurderes feilestimering av prosjektet som lite sannsynlig. I tillegg møtes gruppen med veileder ukentlig, noe som gir oss god kontroll over prosjektets utvikling. Eventuelt kan gruppen i samtale med veileder vurdere hvilke deler av prosjektet som kan utelukkes. Konsekvensen av feilestimering av prosjektet kan være stor, spesielt for oppdragsgiver.

4. For å forhindre at vi bruker for mye tid på mindre deler av prosjektet følger vi en framdriftsplan (figur 3), som kan måtte justeres underveis. Vi anser endringer i tidsplanen som svært sannsynlig ettersom vi kommer til å benytte oss av teknologi som er ny for oss og vi har liten erfaring med systemutviking. Hvis deler av prosjektet skulle avvike betydelig fra planen, noe som må vurderes for hver overskridelse, må veileder og/eller samarbeidsparter kontaktes.

5. Det vil alltid være en risiko for tap av kildekode/data, men gruppens medlemmer har fra tidligere prosjekter god rutine for bruk av versjonskontrollsystemet Git. Med et Bitbucket-repository samt alle tre gruppemedlemmenes lokale installasjoner vurderes risikoen for tap av arbeid som lav. Men det kan ha store konsekvenser for prosjektet dersom det likevel skulle skje at store deler av arbeidet går tapt.

6. Dersom sikkerhetshull i systemet forblir uoppdaget gjennom prosjektet, og det utspiller seg i lekkasje av sensitive opplysninger i pilotprosjektet eller senere, er en av konsekvensene at oppdragsgiver ikke kan bruke systemet. Vi er uerfarne utviklere og har ikke jobbet med personvern før, derfor er det stor sannsynlighet for at systemet ikke er sikkert nok til å behandle sensitive personopplysninger. Konsekvensen av dette er kritisk fordi systemet ikke kan brukes.

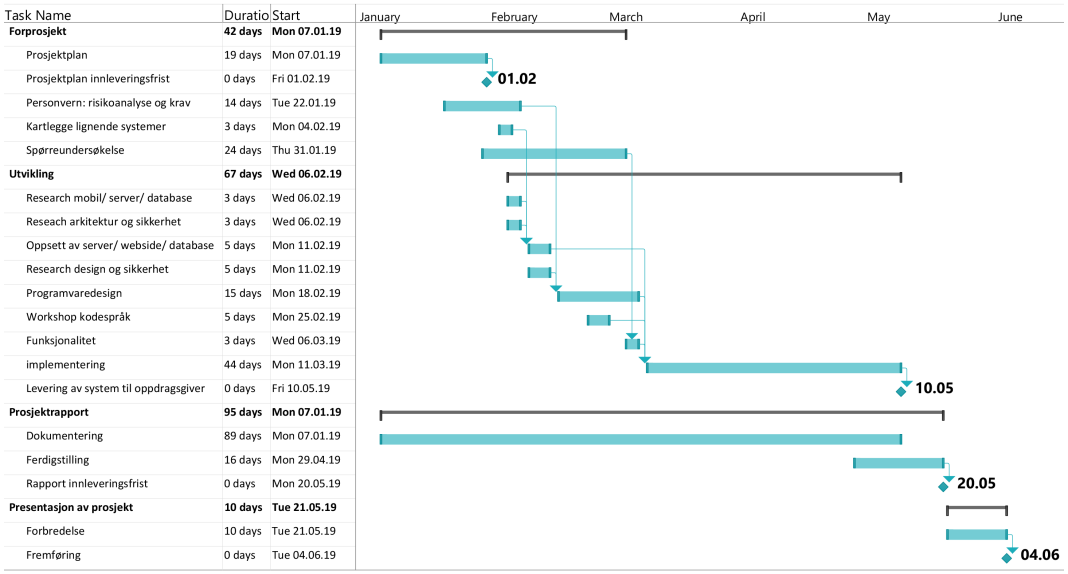
   For å forebygge at dette skjer følger vi datatilsynets veileder *Programvareutivikling med innebgyd personvern* [15]. I starten av prosjektet utfører vi en risikoanalyse av personvern, og sammen med GIT setter vi personvernkrav som vi følger gjennom hele utviklingen.

Vi må etterstrebe god prosess, involvere personvernkravene i hver aktivitet og teste programvaren grundig.

6 Plan for gjennomføring

Vi følger Gantt-diagrammet i utviklingen. Helgedager tas ikke med i beregningen av antall dager vi bruker på hver hendelse, men brukes som buffer ved behov for å sikre fremgang.

6.1 Gantt-diagram



Figur 3: Gantt-diagram

6.2 Liste over aktiviteter (Work Breakdown Structure)

Hendelser	Beskrivelse	Start	Dager
Forprosjekt		**07. jan**	**42**
Prosjektplan	Plan for hvordan prosjektet skal utføres.	07. jan	19
Personvern: risikoanalyse og krav	Risikoanalyse utføres etter datatilsynets veileder for programvareutvikling, og krav settes ut i fra analysen.	22. jan	14
Kartlegge lignende systemer	Finne andre lignende systemer som finnes og se hvordan vi kan skille oss ut fra de.	04. feb	3

Hendelser	**Beskrivelse**	**Start**	**Dager**
Spørreundersøkelse	Planlegge og utføre en undersøkelse for trenere og utøvere, for å avklare hvilken funksjonalitet vi trenger i systemet.	31. jan	24
Utvikling		**11.feb**	**67**
Research mobil/ server/ database	Avklare hvilken teknologi vi skal benytte oss av.	06. feb	3
Research arkitektur og sikkerhet	Research for hvordan systemets kan arkitektur kan designes med tanke på sikkerhet i systemet.	06. feb	3
Oppsett av server/ webside/ database	Server, webside og database settes opp etter valg fra research.	11. feb	5
Research design og sikkerhet	Research for hvordan systemets klassedigram kan designes med hensyn på sikkerhet, og finne løsninger på brukervenning ux-design.	11. feb	5
Programvaredesign	Velger arkitektur og lager klassediagram etter research av design og sikkerhet.	18. feb	15
Workshop kodespråk	Opplæring av kodespråk etter valg av teknologi fra research om mobil/ server/ database.	25. feb	5
Funksjonalitet	Sette opp user stories/ use case etter resultater fra møter og spørreundersøkelse.	06. mar	3
Implementering	Implementering av design og funksjonaliteter i systemet etter valg av programvaredesign og funksjonalitet. Funksjonaliteter kan endres, legges til eller fjernes i denne perioden.	11. mar	44
Prosjektrapport		**07. jan**	**95**
Dokumentering	Dokumentering av prosjektet i henhold til regler beskrevet i avsnitt 5.1.	07. jan	89
Ferdigstilling	Rapporten settes sammen og finskrives.	29. apr	16
Presentasjon av prosjekt		**21. mai**	**10**
Forbredelse	Fremføring lages og øves på.	21. mai	10

Tabell 4: Liste over aktiviter i prosjektet

6.3 Milepæler og beslutningspunkter

Dato	Beskrivelse
01. feb	Prosjektplan innlevering
08. feb	Personvern: risikoanalyse skal være utført og krav satt
08. feb	Valg av teknologi for webside og applikasjon skal være bestemt
15. feb	Webside, server og database skal være satt opp
08. mar	Programvaredesign for systemet skal være ferdig
10. mai	Systemet skal være ferdig utviklet
20. mai	Prosjektrapport innlevering
03. jun	Planlegging av prosjektfremføring skal være ferdig
04. jun	Prosjektfremføring

Tabell 5: Milepæler og beslutningspunkter under prosjektet

7 Verktøy

Listen over verktøy vi bruker i prosjektet vil utvides ettersom vi bestemmer oss for hvilken teknologi vi bruker.

Navn	Type	Bruksområde
Google Disk	Lagringsverktøy for ulike typer dokumenter	Prosjektplanlegging
Overleaf LaTeX	Kompilator og redigeringsverktøy for LaTeX-dokumenter	Prosjektrapport
Toggl	Timeregistrering	Loggføring
Trello	Utviklingsvertøy	Scrumban tavle
Microsoft Prosjekt	Planleggingsverktøy	Systemutvikling
Bitbucket	Versjonskontroll/repository	Konfigurasjonsstyring
SonarQube	Analyseringsverktøy av kildekode	Analyse
Jasmin	Rammeverk for testing av JavaScript-kode	Testing

Tabell 6: Verktøy til bruk i prosjektet

Referanser

[1] *Gjøvik Idrettsmedisinske Team - Hva gjør vi?* Jan. 2019. URL: http://gjovikidrettsmedisin.no/hva-gjor-vi/.

[2] *Personopplysningsloven*. Jan. 2019. URL: https://lovdata.no/dokument/NL/lov/2018-06-15-38.

[3] *Personopplysninger*. Jan. 2019. URL: https://www.datatilsynet.no/rettigheter-og-plikter/personopplysninger/.

[4] *W3Counter: Global Web Stat*. Jan. 2019. URL: https://www.w3counter.com/globalstats.php.

[5] *Javascript Versions*. Jan. 2019. URL: https://www.w3schools.com/js/js_versions.asp.

[6] *Kit Manager*. Feb. 2019. URL: http://www.kit-manager.no/.

[7] *Athlete Monitoring*. Feb. 2019. URL: https://www.athletemonitoring.com/.

[8] *Scrum Guide — Scrum Guides*. Jan. 2019. URL: https://www.scrumguides.org/scrum-guide.html.

[9] Radigan D. *Kanban - A brief introduction*. Jan. 2019. URL: https://www.atlassian.com/agile/kanban.

[10] Straughan G. *From Scrum to Scrumban in 6 Steps + FREE CHEAT SHEET*. Jan. 2019. URL: https://www.youtube.com/watch?v=fgT4AaKcBUA.

[11] *Rules*. Jan. 2019. URL: https://docs.sonarqube.org/latest/user-guide/rules/.

[12] *Jasmine Documentation*. Jan. 2019. URL: https://jasmine.github.io/.

[13] Driessen V. *A successful Git branching model*. Jan. 2019. URL: https://nvie.com/posts/a-successful-git-branching-model/.

[14] T Preston-Werner. *Semantic Versioning 2.0.0*. Jan. 2019. URL: https://semver.org/.

[15] Bergsager D et al. *Programvareutvikling med innebygd personvern*. Jan. 2019. URL: https://www.datatilsynet.no/rettigheter-og-plikter/virksomhetenes-plikter/innebygd-personvern/.

C Møtereferater

Møtereferat

Dato: 10/1-19
Tid: 09:15 – 09:50
Sted: ETC, studievegen 2
Til stede: Terje fra GIT, Dag fra ETC, Johanne, Amund og Cecilie

I desember 2018 hadde vi vårt første møte med GIT hvor vi gikk vi gjennom deres ønsker til systemet. Dette er første møte sammen med støttespiller ETC, som bidrar med teknisk hjelp i prosjektet etter behov.

Sak 1: Fremtidige møter
Vi setter oss et mål om ukentlige møter med varighet på en time i utgangspunktet hos GIT, alternativt hos ETC ved behov og etter avtale. I første omgang settes det opp tre avtalte møter: torsdag 17/1 hos ETC, 24/1 og 31/1 hos GIT hvis ikke annet blir avtalt. Møtene er fra kl. 9-10.

17. januar møtes vi sammen med ETC i studievegen 2 og diskuterer skisser av systemet som hovedtema på dette møtet.

Sak 2: Forventninger
Vi har snakket om forventninger til tidsbruk på rapport og utvikling. Vi burde begynne å skrive på rapporten og dokumentere valg fra starten. Noen uker før fristen, 20. mai, blir mye tid satt av til ferdiggjøring av rapporten.

Sak 3: Krav til systemet
GIT ønsker et system med tre aktører: Helsepersonell, trenere og utøvere.

- Utøverne trenger en applikasjon som er enkel å bruke slik at den faktisk blir brukt. Den må også se bra ut for at utøverne skal ønske å bruke den. Utøverne trenger bare en applikasjon på mobilen hvor de kan rapportere skader eller be om kontakt med helsepersonell. Det er usikkert om utøverne skal ha tilgang på liste over sitt/sine lag.
- Trenerne trenger tilgang til liste over laget de trener. Det er usikkert om de bare trenger mobilapplikasjon eller tilgang til nettleser-grensesnitt i tillegg.
- Helsepersonell trenger tilgang til alt. Det er de som styrer applikasjonen. Vi ser for oss at helsepersonell bruker applikasjonen gjennom en nettleser og får en administrator-rolle i systemet.
- Systemet vil inneholde sensitiv informasjon, og vi må derfor ta hensyn til krav fra Datatilsynet ang dette. Vi må ta kontakt med de eller finne disse kravene.
- Applikasjonen må fenge og brukerne må ville bruke den.
- GITs design profil skal brukes i applikasjonen.

Sak 4: Teknologivalg
I forhold til størrelsen på prosjektet og sikkerheten har ETC kommet med innspill på at det vil være enklest å fokusere på nettsiden, og at mobilapplikasjonen viser nettsiden. Dette vil også hjelpe rundt hvis noen stjeler en telefon så er det verste som kan skje at de kan logge seg på, men det kan også skje på nettsiden uansett.

Applikasjonen må lages for å kunne utvikles videre. Kanskje også integreres sammen med for eksempel journalsystemer som allerede finnes. Mobilapp kan være en mulig utvidelse.

Ved innlogging trengs det ulike tilganger til applikasjonen. Trener har kanskje tilgang til lagets status. Helsepersonell har tilgang til alt. Det tenkes at man kan dele inn innloggingen vanlig innlogging og rød/sikker sone for helsepersonell. Brukernavn og passord, sms med melding tilbake ble nevnt som er mulig løsning, og implementere tredjepartsløsninger som for eksempel Google verifisering.

Hvis vi skal ha en mobilapplikasjon burde den være for både Android og iOS. Rammeverk for applikasjoner på Android og iOS kan være en løsning. Her kan vi gjøre research og spør Tom om tips til rammeverk som er «hot»

nå slik at vi ikke lager noe som blir utdatert. Rammeverket kan brukes til browser og innlogging. Denne blir enkel i første omgang.
Database og server: Stille spørsmål om for eksempel PHP er godt nok i forhold til sikkerhet rundt database og server.

N**este møte 17. januar kl 09:00 hos ETC:**
Til neste møte skal vi sette opp krav til systemet og skisse av funksjonaliteter i applikasjonen som er forståelig for kunden. Dette skal være mer forståelig enn typiske use-case.

Møtereferat

Dato: 17/1-19
Tid: 09:00 – 10:00
Sted: ETC, studievegen 2
Til stede: Terje fra GIT, Dag fra ETC, Johanne, Amund og Cecilie

Sak 1: App for utøver og registrering av skade
Helsepersonell spør ofte om utøveren tror de kan spille kamp, og det er ofte utøveren som vet best selv. Det kan være en mulighet at utøveren kan sette seg selv som grønn/ gul/ rød i systemet, men det er mulig det ikke er relevant at de gjør dette i systemet.
Det er ikke bare skader som burde meldes inn, men også smertetilstander. Dette er for å kunne følge opp og forebygge skader. Registrer skade eller smertetilstand.
I skissene vist frem kan vi fjerne et bilde av personen man kan trykke på, slik at utøveren bare trykker en gang på et slikt bilde. Alternativt en side med foran og bak på personen (for å kunne skille mellom mage og rygg for eks). Her kan vi bruke x og y koordinater for å registrere områder på kroppen. Figuren kan markeres med «her trykket du» for å gi tilbakemelding til brukeren.
Man må kunne gå tilbake ved feil i registrering av skade prosessen.
Vi kan fjerne spillerstatus endring for utøveren – dette kan helsepersonell sette i samarbeid med utøveren.
Et alternativ kan være en «dagbok» på smertetilstand for å kunne følge opp bedre.
Vi må finne ut hvordan smertetilstander skal håndteres i systemet. Smertetilstander kan komme og gå; skal de slettes når smertetilstanden har gått over og registreres som ny smertetilstand hvis den kommer tilbake? Her må vi kanskje finne en bedre løsning.
«Kontakt GIT» burde være «send melding» isteden.

Sak 2: Web app for helsepersonell
Overføring av data: journaler er et eget prosjekt. Systemet burde utvikles med tanke om dette som en mulig utvidelse.
Det må kunne komme inn meldinger fra utøvere.
Beskjeder om nye skader.
Man må kunne gå inn på personsiden til en utøver, ikke bare oversikt.
Statistikk burde kunne hentes inn.

Sak 3: App for trenere
Trener skal ikke legge inn nytt lag i systemet.
Trenere har kun ett lag.
Trenere må kunne legge inn spillere i sitt lag.
Hvilken informasjon utøverne godtar at treneren har tilgang til.

Sak 4: Klubb-rolle i systemet
Vi må se på muligheten å legge inn en ekstra rolle i systemet; klubben. Klubben må kunne si hvor mange lag/grupper de vil ha i systemet.
Klubber kan registrere nytt lag, men dette kan også helsepersonell gjøre.
Alternativt kan helsepersonell gjøre denne jobben.
Vi har ikke et skille mellom trener/ oppmann for et lag. Hvem er admin for en klubb?

Sak 5: Lignende systemer
Det finnes ikke noen særlig gode systemer som brukes i dag etter hva vi har funnet ut av så langt. Ofte er de løsningene som finnes for kompliserte, som fører til at de ikke blir brukt. Vi må lage et system som er helt enkelt for hvordan utøveren bruker systemet. Det kan være at utøveren bare sender inn en melding til helsepersonell i systemet eller legger inn en ny skade/smertetilstand.

Neste møte torsdag 24. januar kl 09:00:
Neste uke hos GIT.

Møtereferat

Dato: 24/1-19
Tid: 09:00 – 10:00
Sted: GIT, ringvegen 2
Til stede: Terje fra GIT, Johanne, Amund og Cecilie

Sak 1: Kvantitative og kvalitative mål
Mulige kvalitative mål:

- Bedre rutiner rundt kommunikasjon der det er taushetsbelagte saker inne i bildet.
- Unngå å måtte bruke sms, mail og facebook-grupper.
- Bedre sikkerhet rundt kommunikasjon rundt sensitive opplysninger
- Forebygge skader på sikt med bedre oversikt
- Redusere skadeoversikt med bedre oversikt
- Hovedmålet: Testløsning som kan tas i bruk. Som kan utvikles videre, og kanskje selge det i markedet
- Pilotprosjekt som man kan utvikle videre og finne ut hva som faktisk trengs i systemet

Mulige kvantitative mål:

- Inntjening i kroner og øre: hva skal det koste? Lisenser for software, eller månedsavgift og antall brukere.
- Inn i et toppnivå-lag i hockey, håndball osv kan være et mål
- Frem til mai: et system som er klar til å tas i bruk som et pilotprosjekt'
- løpet av et år etter det er satt i drift: x antall klubber bruker det

Terje prøver å sette ned noen ord om mål og sender disse på mail.

Sak 2: Trello
Terje lager seg Trello-konto og har fått tilgang til vårt arbeidsbord hvor han kan legge inn ønsket funksjonalitet.

Sak 3: Skisser av systemet og funksjonalitet for de tre ulike brukerne
Helsepersonell:

- Veldig mye rødt, gult og grønt. Ikke like ryddig som skissene for utøver og trenere.
- Helsepersonell kan legge inn mer info og klikke på statistisk data som lagres (se punkt to under utøver ang dette).

Utøvere:

- Når brukeren legger inn/ rapporterer en skade må det være enkelt, mens helsepersonellet kanskje legger inn mer info om skaden og statistisk data.
- Det brukeren legger inn skal fanges opp og kunne brukes til statistikk. Tekst gir ikke statistikk. Systemet kan fange opp data, skadeområde, hvor det skjedde (trening/kamp). Man kan kanskje også se på når brukeren melder inn skader i forhold til når skaden oppsto. Når du logger deg inn på systemet som utøver, skal man ha en egen «min side»? Det er det folk er vant med – en oversikt over alt man har om seg selv i et system.

Trenere:

- Trenere vil ikke bruke tid på å rapportere inn skader og smertetilstander for alle utøverne i laget.
- Trener skal ikke ha tilgang til personopplysninger, men oversikt over spillerstatus.
- Det blir for snevert å bare se lys, man trenger noe mer utfyllende. Kanskje en setning om tilstand, treningsstatus og kamp-klarhets-status.

Vi vil etter hvert finne ut hva som trengs i systemet. Trener og helsepersonell vil se etter hvert hva som er bra og hvilken informasjon som er brukbar og behjelpelig.

Sak 4: Spillerstatus og innmelding av skader

GIT bruker en skala fra 0 til 3, med tekst.

- 0 skadefri (grønn)
- 1 smertetilstand, men kampklar (gul)
- 2 smertetilstand/skade, kan delta kollektiv trening, men kan måtte ha tilpasning, ikke kampklar (oransje)
- 3 skade/smertetilstand, kan ikke trene kollektivt, og ikke kampklar (rød)

Når en utøver rapporterer inn en skade må følgende skje:

- Det må kunne plinge inn hos helsepersonell (trenger kanskje ikke komme hos treneren)
- Helsepersonell kan da umiddelbart trykke rødt, eller en egen farge for dette (se neste punkt), før utøverens skade/smertetilstand er funnet ut av
- Skaden / smertetilstanden / innrapporteringen merkes i systemet som «noe har skjedd, men vi vet ikke omfanget av det er eller hva spillerstatus skal være». Det vil da skille seg ut i systemet istedenfor å legge det inn som rødt lys umiddelbart.
- Etter innrapporteringen er fulgt opp vil statusen bli endret (grønn, gul, orange, rødt eller noe annet).
- Treneren vil kanskje da få en melding. Da kan treneren gå inn og se status og se hva som har skjedd og kunne legge opp treninger og kamper deretter.

Hvem skal ha tilgang til å endre status? – Dette må vi finne ut av.
Spillerstatus får mer faglig tyngde hvis det kommer fra helsepersonell.
Sikkert system som man kan logge seg på er viktig!

Mulige krav til systemet:

- Push varsling
- Skader som er innrapportert, men ikke fulgt opp skal skille seg ut fra andre rangeringer av skader/smerter

Sak 5: Mobil app
En ren web app er presentert som mulig alternativ til å bruke et rammeverk. Denne vil kunne legges til på hjem-skjermen til brukeren og se ut som en vanlig applikasjon.

Annet:
Vi kan søke opp: Fysika – journalsystemer. Kan integreres i framtida. Systemet vårt kan slite med å stå alene for å konkurrere med store omfattende systemer.

Neste møte torsdag 31/2 kl 09:00:

- Vi setter opp en agenda for møtet og sender dette på mail på mandag.
- Terje tenker på noen mål for systemet.

Møtereferat

Dato: 31/1-19
Tid: 09:00 – 10:15
Sted: NTNU i Gjøvik
Til stede: Terje fra GIT, Johanne, Amund og Cecilie

Sak 1: Spørreundersøkelse praktisk
Vi kan sitte i mai med et system som vi er fornøyd med, men uten å vite hva de som skal bruke det vil ha. En pilot kan være uferdig, men det er en fordel at piloten allerede er noe tilpasset de som skal bruke det - også for å få noen til å være med å teste pilotprosjektet. Vi må få til en avtale før vi sender ut spørreundersøkelsen, for å hindre at undersøkelsen blir sett på som "spam". Det kan også bli mer kvalitet på svarene vi får.

Intervjuer kan gjøres over telefon eller møte på kontor. Google skjemaer kan enkelt sendes ut til mange og hente inn statistikk.

Spørreundersøkelsene må ut på et visst nivå. Proff- og semi-proff-nivå må vi ut til. Daglig leder hos disse lagene og trenere er aktuelle å kontakte; kanskje også ned til 16 års-lag. Raufoss fotball er aktuelle å bruke (her har Terje kontakter). Gjøvik Hockey (har ingen avtale, Terje har kontakter), Gjøvik HK, Toten HK (Terje har kontakter), Gjøvik Swans (Cecilie har kontakter) og Storhamar Håndball Damer (eliteserie-klubb). Vi kan forhøre oss litt lenger utenfor Gjøvik, Toten og Land også.

Vi kan bruke Gjøvik Swans som test-spørreundersøkelse-gruppe, og som research til mulige spørsmål?

Sak 2: Spørsmål til undersøkelsen
Trenere har nå bare en oversikt over laget som funksjonalitet i systemet. Vi må finne ut om dette er nok for trenere, eller om de ønsker noe mer fra systemet?
Spørsmål:

- brukes det noe digitalt verktøy?
    - JA: hva fungerer/ ikke fungerer, hva savner du?
    - NEI: kunne du trengt det?

- pris de er villige til å betale for et slikt system?
- Hvordan håndterer laget skader og smertetilstander, bruker laget digitalt system til dette?

Hva spillere trenger, vil ha? Ikke gå rett på funksjonalitet?

- Spillere: gidde å registrere noe med to-faktor-identifisering.

Spørreundersøkelsen burde komme inn i februar/mars-skiftet. Hvor mange lag er nok å spørre? En er alt for lite. Er fem

Terje skriver ned navn, kontaktinformasjon til klubber, og tenker over spørsmål til undersøkelsen.

Bachelorgruppen gjør research på hvordan spørreundersøkelser burde utføres, og tenker ut spørsmål til undersøkelsen. Cecilie tar kontakt med Trenere i Gjøvik Swans og spør de hvordan de håndterer skader.
Alle spørsmål skrived ned i "Forslag til spørsmål" i spørreundersøkelse-mappen på Google Disk.

Sak 3: Arkivere/ slette skader i systemet

Viktigste årsaken til skader er tidligere skader! - tidligere skader får en ny skade. Skader og smertetilstander kan arkiveres istedenfor å slettes. Da kan spilleren gjøre skaden “gyldig” igjen. Reskade.

Sak 4: Nødvendige personopplysninger å lagre
Hvis det ikke er nødvendig å lagre personnummer, vil vi unngå å lagre dette. Vi trenger å lagre personnummer ved utvidelse til journalsystem. Kanskje man trenger å legge inn personnummer fra starten.

Fullt navn, telefonnr, epostadresse og fødselsdato ser vi for oss at er nødvendig.

Helsepersonell har journal-plikt. Rundt dette har vi følgende spørsmål å ta hensyn til:

- Journalen burde egentlig være med i systemet, men blir kanskje for komplisert?
- Skal man overføre informasjon fra et journalsystem til vårt system?
- Skal systemet overta for eksisterende journalsystem?

Vi beveger oss uansett inn i sensitive opplysninger når vi rapporterer inn at en skade har skjedd eller en smertetilstand har oppstått. Legen/ helsepersonel skall kanskje ikke ha mulighet til å legge inn sensitive opplysninger? Vi må finne ut om løsningen må være så tungvint at den ivaretar opplysninger. Hvor langt inn i personopplysninger kan vi bevege oss? Vi må researche nærmere om dette.

Programmet må være konkurransedyktig. Det må bli større i fremtiden. Full journalsystem, følger med krav. Fysika, journalsystem, kan integreres med andre systemer. Kanskje systemet kan integreres her - må til en avtale for å gjøre dette. Systemet kan muligens lages for at det kan integreres med et journalsystem i fremtiden. Det må da kunne integreres, men også kunne stå alene.

Sak 5: Lignende systemer
Kit manager fra Kristiansund. Denne berører en del av det vi jobber med og journalføring. Denne er de fornøyd med i landslaget. Vi kan bestille demo.

Sidelines sports fra Hamar er ikke helt det samme vi holder på med. Aldersbestemte landslag har brukt dette et år og er fornøyde.

Athlete Monitoring er Kanadisk(?) og brukes i Strømsgodset. De er fornøyd med denne og den brukes daglig.

Vi må passe på å ikke kopiere disse.

Hvordan kan vi skille oss fra disse systemene? Noen av de handles om øvelser og program. Det er ikke her vi skal ha fokus. Hovedfokuset vårt er skadeforebygging og behandling av skader. Plukke opp smertetilstander tidlig.

Sak 6: Sikker innlogging
Cisco AnyConnect secure mobility client brukes for å logge inn på fysika.

Raskt og sikkert kræsjer. Vi må finne en mellomting slik at det blir enkelt å bruke systemet, men at det fortsatt er sikkert. Det kan gå utover hva vi kan behandle og lagre av opplysninger i systemet.

Hva kan systemet inneholde uten å bruke to-faktor-identifisering? Skille i rød og grønn sone:
Når en spiller sier “jeg har en skade” er det personopplysninger? trengs det to-faktor? Her må vi gjøre research. Dette vil vi også finne ut av i risikoanalysen av personvern.

Sak 7: Finansiering av pilotprosjekt
GIT selskap som ikke har mange midler. Terje Løkken - enkeltpersonsforetak.

- Søke om midler til pilotprosjekt.
- Prosjektet skal ikke koste mye.

- Server, database, to-faktor-identifisering er det vi ser for oss at vil/ kan koste noe.

Neste møte fredag 8/2 kl 09:00 hos GIT:
Spørsmål til spørreundersøkelse, risikoanalyse og krav til personvern behandling og lagring.

Møtereferat

Dato: 8/2-19
Tid: 09:00 – 09:45
Sted: GIT, ringvegen 26
Til stede: Terje fra GIT, Johanne, Amund og Cecilie

Sak 1: Spørreundersøkelse praktisk
Vi lager skjema, og sender ut til en testgruppe: Gjøvik Swans. Test-spørreundersøkelsen går hovedsakelig først ut til trenere og utøvere. Vi nevner ikke at GIT er med på prosjektet; de står anonyme da de har avtale med testgruppen og vi ikke vil at dette skal påvirke svarene. I neste omgang skriver vi hvem som sender ut undersøkelsen. Ingressen til spørreundersøkelsen skal beskrive hvem vi er og hva svarene blir brukt til. Vi kan kanskje innlede spørsmål eller ingress med: hadde det ikke vært fint å ha et system…

Spørreundersøkelsen burde deles i tre: trenere, utøvere og helsepersonell. Vi må undersøke om disse tre kan kan komme fra samme skjema.

Vi må fange en interesse for å få inn svar/ selge inn spørreundersøkelsen til lagene vi ønsker å sende den ut til. Kontaktperson må informeres slik at de kan informere videre og oppfordre utøvere og trenere til å svare. Noen av klubbene/lagene kan ses på som potensielle samarbeidspartnere og mulig deltagere i pilotprosjekt.

Vi ønsker å få svar på hvor mye det syndes rundt personvern/ deling av spiller-opplysninger, og om lagene bruker noen form for digitalt verktøy i forbindelse med skader/smertetilstander (kommunikasjon og oversikt over spillere). Generelt vil vi ha svar på hva trenere og utøvere ser etter i en sånn app, hva skal til for at de skal bruke den, hva skal en slik app inneholde for å være verdt noe for de og om de trenger et slikt system.

Sak 2: Spørsmål til spørreundersøkelsen
Vi tenker at trenere kan tåle mer spørsmål i undersøkelsen. Vi tenker også at et åpent spørsmål kan være hensiktsmessig, hvor deltakerne skriver åpent svar i et tekstfelt. Dette spørsmålet kan være hva de kan tenke seg i et slikt system, eller hva de ville forbedret med det de bruker i dag. Resten kan være avkrysning med noen åpne annet-felt.

Vi trenger ikke spørre hva trenere trenger av oversikt over spillerne. Det er en selvfølge at de trenger oversikt over hvem som er skadet, hvem som kan trene og hvem som kan spille kamp. Vi må spørre om hvordan man har oversikt og kommunikasjonen rundt skader. Vi må også minne de på hva vi snakker om: kommunikasjon utenom treninger og kommunikasjon rundt skader/ smertetilstander.

Vi må spørre trenere om det finnes et digitalt verktøy de bruker. Hvordan disse fungerer, hvordan de blir brukt, hva som lagres og hvordan kommunikasjonen rundt skader foregår - fra en skade/smertetilstand oppstår, treneren og helsepersonell får vite om det, oppdateringer om tilstanden på skaden/ smertetilstanden og om det føres noe historikk og statistikk. Vi må også spørre om hva de liker og hva de ikke liker med de digitale verktøyene.

Vi må finne ut om vi skal ha direkte spørsmål og hvordan vi skal stille spørsmålene. Hvordan kan man forebygge og redusere skadetid. Vi tenker det er å sette det i system med oversikt, historikk og statistikk. Vi må spørre hva de tenker om dette.

Spørsmål til utøvere kan være om hvordan de kommuniserer med trenere og helsepersonell rundt skader/ smertetilstander: Har du hatt en skade siste året: hvordan meldte du i fra om dette. Hvordan ble det fanget opp av helsepersonell, hvordan informerte du treneren?

Kommunikasjon er noe utenfor oppgaven, men det må være kommunikasjon mellom utøver og helsepersonell. Vi må spørre utøvere om hvordan de ønsker dette skal være.

Vil vil prøve å unngå at spillere ikke vil rapportere skader/ smerter fordi de er redde for å ikke kunne spille kamp. Systemet skal ikke skremme til å rapportere, men oppfordre til å gjøre det for å kunne forebygge skader.

Sak 3: Sikre soner i systemet
Vi snakker med Dag om hvordan dette kan deles opp. Vi har definert rød og gul sone, og satt opp forslag til oppdeling i prosjektplanen.

Bank-ID og Bank-ID på mobil. Ser ut til å være gratis å bruke. Vi har testet hvor enkelt dette er i bruk. Dette varierer, noen steder er mobilnr og fødselsdato lagret, og noen steder må du skrive inn begge deler før du får logget inn.

Utøver sender et signal om at det har oppstått en skade eller smertetilstand. Dette er en personopplysning og må ligge i rød sone. En mulighet er å sette det trener og helsepersonell gjør i rød sone, mens utøveren er knyttet til et meldingssystem for å bare kontakte helsepersonell. Skadehistorikk og detaljer må sikres, mens pushmeldinger til timer kan være utenfor sikker sone, og kanskje også melding om “noe har oppstått”.

Det skal være enkelt for utøveren, men så fort det handler om opplysninger som ikke skal deles må det være sikret. Systemet kan utelate to-veis kommuniksajon mellom utøver og helsepersonell, og isteden være et varslingssytem til helsepersonell.

Pushvarsler om timer (bekreftelser) og pushvarsler med personinfo hvor man må logge seg inn med to-faktor for å sende og skrive mer info kan være en løsning. Helsepersonell kan få et pushvarsel, men ser når de åpner appen om de må logge seg inn videre for å lese hva det handler om.

Ekstra:
Mulig sparringspartner: kunnskapsparken.

Neste møte torsdag kl 09:00 – 10:30 hos GIT
- Spørreundersøkelse gjennomgang og godkjenning før utsendelse til testgruppe (hvis dette ikke gjøres før)
- Timeplanlegging og påminnelser neste møte. Push varsler.

Møtereferat

Dato: 14/2-19
Tid: 09:00 – 09:45
Sted: GIT, ringvegen 26
Til stede: Terje fra GIT, Johanne og Cecilie

Sak 1: Spørreundersøkelse testgruppe
Vi har gått gjennom spørreundersøkelsen, og satt opp endringer som må gjøres før vi sender undersøkelsen ut til testgruppen. Endringene er:

- Spørsmål til utøvere
    - Mer info om appen i spørsmålet om hvor mange felt utøveren "gidder" å fylle ut. Infoen kan være noe sånt som spørsmålet under: "Tenk deg at du bruker vår app…."
- Spørsmål til trenere
    - Hvordan trenere har oversikt over treninger og kamper (fire spørsmål)
        - Vi slår sammen svaralternativene "spredt … og i hukommelsen". Dette er det samme.
        - Digitalt verktøy som excel, word, osv
        - Legge til et "Andre"-felt de kan fylle ut. Vi vil få ut hvilke verktøy som finnes
    - Status (i flere spørsmål)
        - Her må vi være tydeligere på hva vi mener med status. Vi kan sette det inn i introen slik at det ikke blir oversett som en beskrivelse på et av spørsmålene.
    - Hva treneren trenger
        - Legge til "Andre"-felt
    - Spørsmål om hva treneren er fornøyd med/ savner i digitale verktøy de bruker i dag kan flyttes opp under spørsmål om hvilke verktøy treneren bruker.
    - Spørsmål om hvilke opplysninger trenere synes det er "greit" å motta kan omformuleres til hva som er nyttig for treneren å ha. Dette kan da bli et dobbelt spørsmål, da må vi vurdere å fjerne det.
    - Legge til spørsmålene fra helsepersonell-delen ang redusering av skadetid og skadeforebygging
- Spørsmål til helsepersonell
    - Spørsmål om kommunikasjon med klienter, og kollegaer ang klienter
        - Formulering: Hvordan kommuniserer du med utøvere og trenere?
        - Formulering: Hvordan kommuniserer du med dine kollegaer ang skader/ smerter til utøvere?
        - Legge til "Andre"-felt
    - Gradering fungerer fint med liten grad/ stor grad
- Generelt
    - På testkjøringen av undersøkelsen kan vi ha flere "Andre"-felt. Testgruppen er liten som gjør at det ikke blir tungt å gå igjennom disse svarene, og vi kan få frem punkter eller nye spørsmål ved å få flere åpne tanker fra de
- Intro til spørreundersøkelsen
    - Fjerne beskrivelsen om at de som er trenere og utøver må svare på undersøkelsen to ganger. Dette kan vi fortelle til kontaktpersonen i testgruppen
- Spørsmål om spørreundersøkelsen
    - Vi ser det som nyttig og også spørre testgruppen om hva de synes om spørreundersøkelsen. Dette gjør vi med to spørsmål:
        - Et spørsmål med flere mulige avkrysninger: uklare spørsmål, rare eller dumme spørsmål, vanskelige å forstå, virkelighetsfjerne spørsmål og om skjemaet fungerte teknisk og praktisk.
        - Et åpent spørsmål om kommentarer til svar fra forrige spørsmål og annet de vil kommentere.
    - Spørsmål om hvilke opplysninger om skader som legges inn må omformuleres. Det må komme bedre frem at dette er ved siden av et journalsystem

Spørreundersøkelsen gjøres ferdig i dag, torsdag 14. februar, og sendes ut senest i morgen, fredag 15. februar, til kontaktpersonen i testgruppen. Vi må få kontaktpersonen til å dele lenken til spørreundersøkelsen, og informere om at spørreundersøkelsen er en testrunde, at de som er trenere og utøvere tar undersøkelsen to ganger. Kontaktpersonen burde også oppfordre testgruppen til å svare på undersøkelsen han har sendt ut de neste treningene før fristen.

Vi sender også lenken til Terje, som videresender denne til en gruppe helsepersonell.

Fristen for svar på spørreundersøkelsen er satt til onsdag 13. februar. Vi får da tid til å se igjennom svarene og kan ta opp dette på neste møte med GIT.

Sak 2: Spørreundersøkelse andre runde
Svarfrist for spørreundersøkelsen må være en uke før 5. mars slik at vi får tid til å gå gjennom svarene.

Sak 3: Databehandleravtale
Det må lages en databehandleravtale mellom behandlingsansvarlig og databehandlere. Denne avtalen er noe som skal være på plass ved kjøring av pilotprosjektet. Vi lager et utkast av denne.

Neste møte torsdag 21. februar 09:30 hos GIT

- Vi går gjennom svarene fra spørreundersøkelsen og diskuterer endringer vi må gjøre før vi sender den ut til idrettsklubber
- Legger en plan for distribuering av spørreundersøkelsen

Møtereferat

Dato: 21/2-19
Tid: 09:30 – 11:00
Sted: GIT, ringvegen 26
Til stede: Terje fra GIT, Johanne, Cecilie og Amund

Sak 1: Spørreundersøkelse endringer

- Endre spørsmålet: “Hvor godt tror du ditt personvern ivaretas angående taushetsbelagt informasjon?” Vi må få inn at det handler om digitale verktøy. Dette spørsmålet hjelper ikke oss for hva vi trenger i systemet, og kan evt droppes.
- Innledningen trenger noe omformulering. Lege byttes ut med helsepersonell i teksten. I tillegg ble noen mindre endringer gjort på møtet.
- Fjerne spørsmål om spørreundersøkelsen.

Sak 2: Spørreundersøkelse utsending
Bestemme hvem vi skal kontakte

- Raufoss fotball, proffnivå, Terje har kontakt her gjennom kollega
- Toten HK, nasjonalt, semiproff, Terje har kontakt
- Storhamar håndball, Terje har kontakter
- Hamar fotball damer, elite-serie, Amund har en kontaktperson.
- Gjøvik hockey, nasjonalt nivå, semiproffe, Terje har kontakter her.
- Gjøvik HK,
- Bergen?
- Vi tenker over mulige kontakter i andre byer/landsdeler.

Vi vil ha svar fra minimum tre klubber. Vi må ringe/ snakke med trener eller helsepersonell, ikke sende mail. Vi er da ute etter å få lov til å gjennomføre en spørreundersøkelse. Etter godkjenning sender vi ut en mail med link til undersøkelsen som de vi kontaktet sender ut til laget. Det er viktig å få de til å informere på treninger, for å få høy svarprosent.

I første omgang:

- Terje ringer Raufoss fotball, Toten HK og Storhamar håndball
- Amund ringer Hamar fotball damer

Vi går videre på lista hvis vi ikke får sende ut undersøkelsen hos disse fire.

Fristen for å sende ut bør være onsdag 27. februar
Fristen for å få svar på undersøkelsen er innen 5. mars.

Sak 3: Pilotprosjektet
Git søker om midler. Spørreundersøkelsen hjelper på å søke om midler, vise til interesse og et behov.
Vi burde legge opp til at det er mulig å legge systemet til som en modul i et større system, mulig journalsystem.

Reelle personopplysninger må beskyttes skikkelig under pilotprosjektet. BankID er en stor kostnad for GIT slik situasjonen er i dag. Google autentisering kan brukes som en midlertidig autentisering, som kan byttes ut med BankID på sikt. Systemet blir da utviklet slik at det skal være enkelt å legge til ny/ endre autentiseringsmetode.

Sak 4: Testing av systemet
Terje bør også få tilgang til testmiljøet. Vi snakker med Eigil om dette er mulig i testmiljøet i SkyHiGh.

Neste møte fredag 01.03 kl 08:30 hos GIT
Status spørreundersøkelse, vi viser frem hva vi har jobbet med (visuelt system)

Møtereferat

Dato: 1/3-19
Tid: 08:30 – 11:00
Sted: GIT, ringvegen 26
Til stede: Terje fra GIT, Johanne, Cecilie og Amund

Sak 1: Spørreundersøkelse
Gjøvik Hockey, Storhamar håndball er med og har fått undersøkelsen..
Fotballaget Fart på Hamar - avventer svar.
Raufoss fotball har ikke fått spørreundersøkelse, avventer svar.
Fristen er litt kort (5. mars), de som har fått spørreundersøkelsen får ut neste uke (10. mars).

Sak 2: Domenemodell
Helseteam, helsepersonell.

Klubb har flere lag,

Gjøvik hockey har ulikt helsepersonell, de har ikke fast avtale med et helsepersonell. Hvordan skal spillerstatusen. Hvordan legges informasjonen inn i spillerstatus og om.

Treffe behovet i markedet: for de som har avtale med HP og de som bruke ulike HP. Skille mellom to moduser (med og uten fast helsepersonell). Trener kan legge inn informasjon om spillerne sine for å ha oversikt over spillerne uten å ha avtale med et helseteam. Eliteklubber har kanskje en fast klubb de bruker, og pålegger sine spillere å bruke det teamet?

Begynne med den helt enkle modellen med trener, utøvere og en fysioterapaut.

Helseteamet har tilgang til journal-notatene til alle i det teamet.

Vi ser vekk fra at helsepersonell er ulike utenfor et team.

Lagring av data i database, sikkerhet. Dele opp informasjonen i tabeller eller databaser.

Sak 3: Navn, logo
Vi bruker fargene og logoen til GIT under bachelorprosjektet. Systemet heter Spillerlogistikk.

Neste møte avtales med Terje og Dag over mail.

Møtereferat

Dato: 21/3-19
Tid: 09:00 – 10:00
Sted: GIT, Ringvegen 26, Gjøvik
Til stede: Terje, Amund, Johanne og Cecilie

Sak 1: Applikasjonen - frontend
Forsiden: Her kan man enten ha en logo som tar stor plass, eller valg om å logge inn eller registrere ny bruker.
Registrering: Vanlig bruker. Hvor skal reg for helsepersonell, trenere og utøvere være?
Min side: Mangler spillerstatus info eller en merknad

Lagoversikt. Vi trenger en statistikk side med skaderate (hvor mange spillere har treneren klar til en hver tid i løpet av året, i forhold til antall spillere, prosent skaderate).

- Hvis det bare er et lag (for trener for eksempel) burde man ikke måtte "åpne" den raden, den burde være åpen når man kommer inn på siden.

Footer: Personvernerklæring, og om appen. Brukeren skal hele tiden ha tilgang til personvernerklæringen, og det er fint at den ligger tilgjengelig her.

Sak 2: Funksjonalitet for laget og statistikk
Lag-raden: kjernen for oversikt over laget og status

- antall spillere tilgjengelig
- antall spillere ute (med spillerstatus (2, 3)

	dato	dato	dato
antall spillere tilgjengelig	0	10	0
antall spillere ute	0	1	0
Skaderate i %	0	10%	0

Trener må se skaderaten før treninger og kamper. Det skal noteres punkter et fast tidspunkt f. eks hver dag kl 23:59.

Utøvere-radene: kjernen for oversikt over spillerstatuser

- draktnummer på spillere
- navn på spiller
- status farge

Skal det være mulighet for å sortere kolonnene med spillere? Isåfall kan vi sortere etter skade eller tilgjengelighet. Hvis man vil se noe mer kan man klikke seg inn på en side med mer informasjon, hvor man finner merknader/ spillerstatus info.

Sak 3: Innlogging, sikker sone
Logge inn med en basistjeneste: Er det noe vi kan dra utenfor den sikre sonen? Vi vurdere dette.

Sak 3: Spørreundersøkelsen
Vi har per dags dato bare fire svar fra undersøkelsen, og rundt 30 svar fra testundersøkelsen. Vi bruker svarene fra testrunden. og bruker ikke mer tid på å få flere svar fra andre lag.

Sak 4: Design, fargevalg, logo
Blåfargen passer bra til spillerstatus fargene (grønn, gul, orange, rød)

Videre: Lagliste og oversikt med funksjonalitet. Vi legger inn fiktive lag og utøvere for å teste.
Neste møte avtales etter behov over mail.

Møtereferat

Dato: 4/4-19
Tid: 09:00 – 10:00
Sted: GIT, Ringvegen 26, Gjøvik
Til stede: Terje, Amund, Johanne og Cecilie

Sak 1: App: Utøver-registrering
Når man registrer seg skal man ikke ha tilgang til systemet uten å være godkjent. Utøveren burde kunne logge seg inn etter han er godkjent av trener, klubb-representant eller admin.

Per dags dato er registreringen laget slik at utøveren velger det laget han tilhører slik at forespørselen går til riktig lag. Utøveren burde kanskje ikke se hvilke lag som bruker systemet. Alternativer til å hindre dette er:

- Legge inn alle i laget og utelukker at utøveren kan registrere seg selv
- Utøveren skriver inn klubben man hører til i stedet for å velge under registrering
- Når en utøver registrerer seg kan de skrive inn et kodeord som hører til idrettslaget (som trener eller klubb-representant har laget og delt med utøverne)

Sak 2: App: Innlogging
To-trinn bygges på etter hvert. Vi implementerer Google-verifisering istedenfor bankID, men legger opp til at man kan bytte til bankID under pilotprosjektet hvis det blir ønskelig.

Sak 3: App: Godkjenning av utøvere

- Avviste spillere trenger ikke ligge lett tilgjengelig under lagoversikten.
- En må ha administratortilgang til laget, det burde bestemmes når et lag/helsepersonell blir kunde av systemet. Evt kan både trenger, klubb-representant og helsepersonell ha administratortilgang.
- Når en spiller slettes, kan man ringe support og få dette fikset. Vi trenger ikke lage funksjonalitet for å fikse dette.
- Nye forespørsler må vises som varsler.

Sak 4: Lagoversikt
Endringer i lagoversikt:

- Når du trykker på lagoversikt, lukkes alle åpne checkbox-er.
- Her må man kunne gå inn på en egen side for laget, etter inngangsportalen med forespørsler (lagoversikt).
- Siden for hvert lag skal inneholde statistikk, og liste over spillerne (slik som excelarket Terje har vist oss).
- Hvis en bare har et lag, bare gå inn på laget.
- Forespørsler skal ikke ligge inne på siden med oversikt over et lag.

Sak 5: Oversikt over spillerne på et lag
Vi skal ha med kolonnene: spiller-nr, navn, fødselsdato, status(0-3, farger og tall inni, oppdatert til enhver tid)
Når du trykker inn på spilleren får du mer informasjon om spilleren og spillerstatusen (Helsepersonell skal kunne administrere denne siden):

- navn, type skade, journal, spiller-logg, evt når spilleren er forventet tilbake

0: skadefri
1: liten alvorlighetsgrad, kampklar
2: moderat alvorlighetsgrad, under rehab, kan delta på trening, ikke kampklar
3: større alvorlighetsgrad, rehabilitering, kan ikke delta på kollektiv trening, ikke kampklar

Neste møte: Slutten av neste uke; torsdag eller fredag.
Vi setter ned noen mål til neste gang, og avtaler møte og mål på mail.

Møtereferat

Dato: 11/4-19
Tid: 10:00 – 09:00
Sted: GIT, Ringvegen 26, Gjøvik
Til stede: Terje, Amund, Johanne og Cecilie

Sak 1: Registrering av nye brukere
Vi jobber videre med å knytte brukere opp mot et lag ved å bruke en invitasjonskode.

Sak 2: Registrering av nye skader
"Hva er skaden din?" er kanskje ikke den beste måten å spørre brukeren om en tittel på den nye skaden, men vi lar denne stå inntil videre.
Det kan være hensiktsmessig å legge inn en type skade (akutt, smertetilstand o.l), men vi lar det være enklest mulig nå. Vi lar også valg om ønske om å bli kontaktet for time stå.

Når en bruker legger inn en ny skade, settes spillerstatusen til grå (-1), som betyr noe av uavklart/ ny skade som helsepersonell ikke har sett på.

Sak 3: Melding om ny skade til helsepersonell
Helsepersonell skal få varsling når en skade er meldt inn. Vi velger selv om meldingene skal ligge på hver skade det handler om, eller en felles for hver utøver.

Sak 4: Trenerens oversikt over laget
Treneren skal bare få varsel når spillerstatus til en av utøverne endres (0-3), men ikke når en ny skade legges inn. Trener skal allikevel kunne se på oversikten over laget at utøvere har grå (-1) spillerstatus.

Trener skal ikke ha tilgang til meldingsutveksling på samme måte som utøvere og helsepersonell, men skal få en generert melding/ varsel når en spillerstatus endres.

Sak 5: Prioriteringer videre
Vi prioriterer å få inn funksjonalitet rundt hovedpoenget med applikasjonen; oversikt over spillerstatus. Skaderapportering kan være enkel slik den er nå. Design kan også være enkelt.

Vi legger også inn hovedsider for trenere og helsepersonell. Og legger inn skiller i applikasjonen mellom de tre ulike bruker-typene.

Neste møte
onsdag 17. april. Tidspunkt avtales på mail.

Møtereferat

Dato: 17/4-19
Tid: 13:30 – 14:30
Sted: GIT, Ringvegen 26, Gjøvik
Til stede: Terje, Amund, Johanne og Cecilie

Sak 1: Registrering av nye brukere
Vi har laget registrering av brukere med invitasjonskode, og går for denne løsningen.

Sak 2: Oversikt over utøvere i et lag
Tabellen med utøvere i et lag er laget med mulighet for å sortere etter flere enn ett kriterie. Vi beholder denne funksjonaliteten. Listen er sortert på spillerstatus når du går inn på siden, og vi legger til sortering på etternavn som sorteringskriterie nummer to.

Det er mulig det er hensiktsmessig at vi legger inn posisjonen til spilleren. Vi legger inn dette som tilleggsinformasjon under "vis detaljer". Dette blir ikke prioritert først.

Vi legger til en kolonne som erstatter spillerstatus "-1". Denne skal inneholde om spilleren har skader som helsepersonell ikke har sett på/ er ubehandlet (se sak 3), slik at helsepersonell også kan sortere listen etter dette kriteriet.

Oppdragsgiver er fornøyd med at tabellen over utøvere er det første som kommer opp etter helsepersonell har logget seg inn.

Sak 3: Spillerstatus
Spillerstatus skal kun settes på utøveren som en helhetsvurdering. Vi legger inn ny kolonne i databasen, og fjerner denne kolonnen fra hver skade, og oppdaterer logikken deretter.

Sak 4: Inbox
Hver skade har en inbox, som blir lagt til når en ny skade registreres. En mulig endring er å legge inn hver inbox i skade-oversikten for utøvere, slik at det ikke oppleves (og kan misbrukes) som et chat-rom. Vi lar inbox-en være slik den er nå, for å gi mulighet til å avtale timer og enkelt gi mer informasjon om hver skade til helsepersonell. Hvordan dette fungerer i praksis må testes ut i pilotprosjektet.

Sak 5: Push varslinger
Når en utøver har lagt inn en skade er det viktig at helsepersonell får beskjed om det. Vi jobber videre med å legge inn push varsling. Push varslingene gi noe informasjon om varslingen gjelder ny skade, ny melding eller lignende. Det er viktig at brukerne også kan se om mottakeren av meldingen har sett meldingen. Helsepersonell vil gjerne vite om treneren har sett en ny statusendring. Det kan være hensiktsmessig å sende med navnet til hvem som har endret spillerstatus på en utøver, til treneren.

Videre:
Vi prioriterer å legge inn push varslinger, to-faktor-autentisering og en utøver-side for helsepersonell.

Neste møte
torsdag 2. mail 09:30 hos GIT.

Møtereferat

Dato: 2/5-19
Tid: 09:30 – 10:20
Sted: GIT, Ringvegen 26, Gjøvik
Til stede: Terje, Amund, Johanne og Cecilie

Sak 1: Push varslinger
Vi legger inn:

- push varslinger ved ny skade til helsepersonell
- push varslinger til trener ved endring av spillerstatus

Sak 2: Spillerside for helsepersonell
Vi legger inn:

- Endring av spillerstatus
- Personnummer
- (Hvis tid) Redigerbar behandlingsplan som gjelder for utøveren som helhet

Helsepersonell burde ha mulighet til å legge til nye skader for en utøver. Dette får vi ikke tid til.

Sak 3: Treneren
Vi legger inn en forenklet versjon av tabellen over spillere helsepersonell har, med navn og spillerstatus, og statistikk over skaderate og antall tilgjengelige spillere til kamp.

Sak 4: Meldinger
Vi har en meldingstråd for hver skade mellom utøver og helsepersonell, og en meldingstråd for hver utøver mellom utøver, trener og helsepersonell.

Sak 5: Spiller-tabell
På tabellen kan man legge til at man kan trykke på hele raden for å få opp en side for den utøveren hvor man kan endre spillerstatus. Da slipper man å flytte hele siden på mobil for å kunne trykke på “vis detaljer” og så trykke seg inn på spillersiden til utøveren.

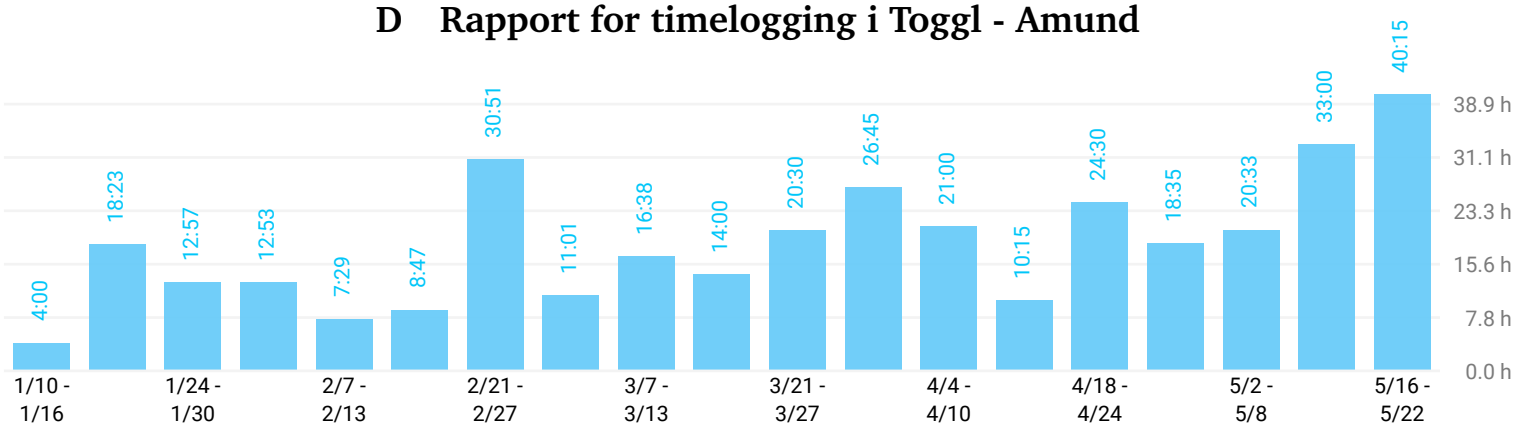
Videre:
Vi prioriterer endring av spillerstatus for helsepersonell, gjør ferdig push varslinger av meldinger og skader og legger inn tabell over utøvere på trener-siden.

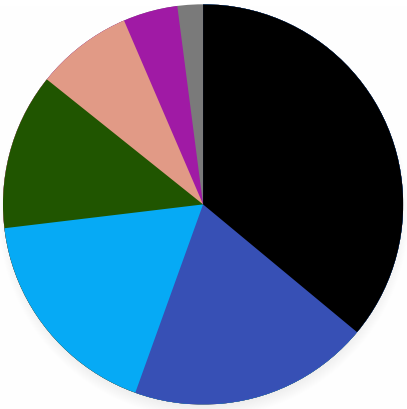
Summary Report

January 10, 2019 – May 19, 2019

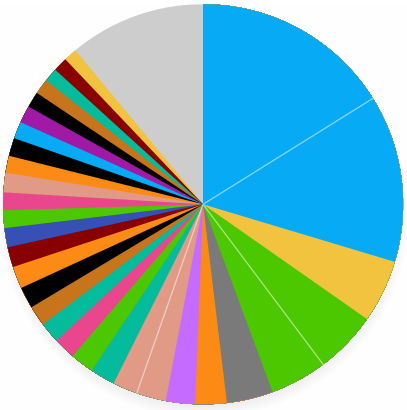
TOTAL HOURS: 352:24:52

D Rapport for timelogging i Toggl - Amund





PROJECT	DURATION
Koding	126:53:00
Rapport	68:45:00
Research	61:48:21
Prosjektplanlegging	44:43:31
Webdesign	27:30:00
Møter	15:45:00
Without project	7:00:00



TIME ENTRY	DURATION
Rapportskriving	56:45:00
Koding	47:35:00
JS - Codecademy	18:00:49
Utøverside	17:48:00
Database	16:00:00
Without description	13:30:00
Møte med Terje	8:45:00
Design utøverside	8:30:00
Div	8:00:00
PWA research og Google PWA training	7:16:00
Database og testdata	7:00:00
Prosjektplan og bestemt systemutviklingsmodell	6:53:10
Research GIT mm.	6:08:03
Database og triggere	6:00:00
Webcomponents	6:00:00
Registrering UI	6:00:00

Skjema og meny	6:00:00
Google Polymer PWA	5:45:00
JS og CSS - Codecademy	5:27:00
Konfigurasjonsstyring	5:19:05
PWA starter kit	5:10:49
Organisering og kvalitetssikring	5:08:59
Grid	5:00:00
Prosjektplan, research	5:00:00
Reasarch - patterns, PWA, JS	5:00:00
Eksisterende lignende produkter	4:45:00
Database - binary logs	4:30:00
Møte med Tom	4:30:00
Prosjektplan + div	4:00:00
Skjema	4:00:00
Ferdigstilling prosjektplan	3:45:00
Other time entries	38:52:57

PROJECT - TIME ENTRY	DURATION
Koding	126:53:00
Database	16:00:00
Database - binary logs	4:30:00
Database og testdata	7:00:00
Database og triggere	6:00:00
Div	6:00:00
Div knoting	3:15:00
Grid	5:00:00
Klasser - brukere	0:45:00
Koding	47:35:00

PROJECT - TIME ENTRY	DURATION
Konfigurering Linux	2:30:00
Registrering - telefonnummer- og passordkrav	2:30:00
Testdata til database og stored procedures	1:30:00
Utøverside	17:48:00
Without description	6:30:00
● Møter	15:45:00
Møte med Terje	8:45:00
Møte med Terje og Dag	1:00:00
Møte med Terje og Dag	1:30:00
Møte med Tom	4:30:00
● Prosjektplanlegging	44:43:31
Ferdigstilling prosjektplan	3:45:00
Konfigurasjonsstyring	5:19:05
Mockup og funksjonalitet for bruker	3:30:00
Oppsett Trello + div	0:59:14
Organisering og kvalitetssikring	5:08:59
Planlegging, diskusjon	0:30:00

PROJECT - TIME ENTRY	DURATION
Prosjektplan + div	4:00:00
Prosjektplan og bestemt systemutviklingsmodell	6:53:10
Prosjektplan, research	5:00:00
Research GIT mm.	6:08:03
Research, systemutviklingsmodeller	1:00:00
Researh PWA	2:30:00
● Rapport	68:45:00
Rapportskriving	56:45:00
Struktur rapport	3:00:00
Use Case diagram	2:30:00
Use Case diskusjon	0:30:00
Webcomponents	6:00:00
● Research	61:48:21
CSS - Codecademy	2:36:06
Div	2:00:00
Eksisterende lignende produkter	4:45:00
Google Polymer PWA	5:45:00

PROJECT - TIME ENTRY	DURATION
HTML og JS - Codecademy	2:00:00
JS - Codecademy	18:00:49
JS - Condecademy	0:54:00
JS - Interactive Websites, Codecademy	0:16:48
JS og CSS - Codecademy	5:27:00
PWA	1:31:49
PWA research og Google PWA training	7:16:00
PWA starter kit	5:10:49
Reasarch - patterns, PWA, JS	5:00:00
Responsive Web Dev - Codecademy	1:05:00
● Webdesign	27:30:00
Design utøverside	8:30:00
Login-box design	3:00:00
Registrering UI	6:00:00
Skjema	4:00:00
Skjema og meny	6:00:00
Without project	7:00:00

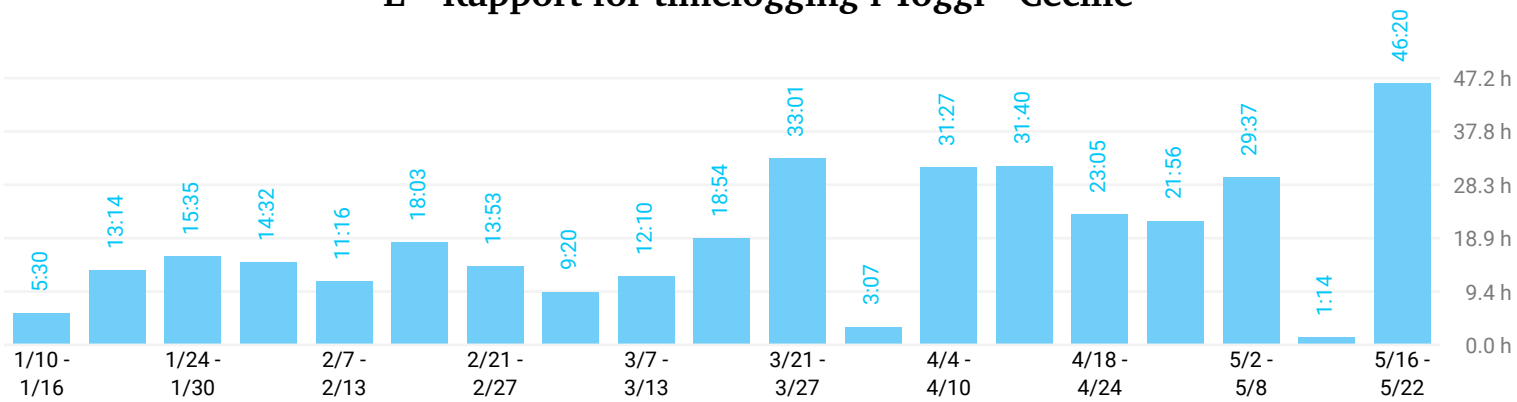
PROJECT - TIME ENTRY	DURATION
Without description	7:00:00

Summary Report

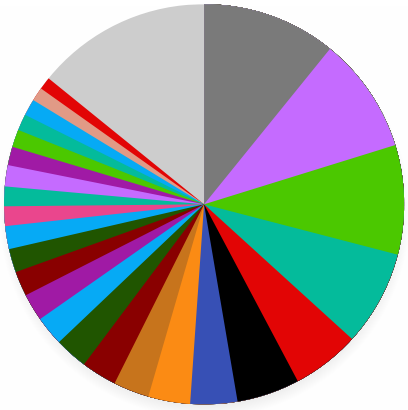
January 10, 2019 – May 19, 2019

TOTAL HOURS: 354:02:29

E Rapport for timelogging i Toggl - Cecilie



PROJECT	DURATION
Koding	182:30:29
Prosjektplanlegging	51:38:58
Research	50:42:13
Rapport	48:22:21
Møter	20:48:28



TIME ENTRY	DURATION
Without description	38:30:18
Push varslinger	33:07:28
Uleste meldinger	31:05:46
Meldingsbox	27:26:46
Prosjektplan	19:23:19
Redux	17:55:06
Elementer	13:19:38
Møte med GIT	12:00:53
Godkjenning av utøvere	10:30:29
Domenemodell	9:54:18
Legge til skader	9:20:06
JavaScript	8:25:06
Service workers	7:50:43
Node.js	7:27:38
Research teknologi	6:37:07
Oppdatering på prosjektet og login funksjon	6:22:22

Spørreundersøkelse	5:50:48
Skaderapportering	5:41:30
Prosjektplan, aktiviteter og sikkerhetskrav	5:40:33
Rapport	5:26:15
polymer PWA	4:56:19
Frontend	4:46:36
pc setup	4:41:00
Godkjenning av brukere og legge til skader	3:53:33
Arkitektur	3:50:22
Other time entries	49:58:30

PROJECT - TIME ENTRY	DURATION
● Koding	182:30:29
Elementer	13:19:38
Frontend	4:46:36
Godkjenning av brukere og legge til skader	3:53:33
Godkjenning av utøvere	10:30:29
JavaScript	1:25:16
Legge til nye lag	0:18:44
Legge til skader	9:20:06
Login	1:58:48
Meldingsbox	27:26:46
Node.js	7:27:38
Oppdatering på prosjektet og login funksjon	6:22:22

PROJECT - TIME ENTRY	DURATION
Push varslinger	33:07:28
Redux	17:55:06
Service workers	7:50:43
Skaderapportering	5:41:30
Uleste meldinger	31:05:46
● Møter	20:48:28
Møte med GIT	12:00:53
Møte med GIT og ETC	1:00:00
Møte med veileder	2:35:37
Møte med veileder og møte med gruppa	1:00:00
Møte med veileder og skriving av møtereferat	1:10:00
Møte veileder, møtereferat	0:28:08
Møtereferat	2:33:50
● Prosjektplanlegging	51:38:58
Gjennomgått utviklingsmodeller	1:00:00
Jobbet med Prosjektplan	2:00:00
Mockup skisse for trenere, helsepersonell. Latex oppsett. Jobbet med Prosjektplan	3:30:00

PROJECT - TIME ENTRY	DURATION
Møtereferat, rapportfiksing	2:13:15
Møtereferatskriving	0:38:49
pc setup	4:41:00
Prosjektplan	19:23:19
Prosjektplan, aktiviteter og sikkerhetskrav	5:40:33
Prosjektplan, møtereferat, research	3:04:00
Prosjektplan: utviklingsmodell og gantt	3:20:00
Renskriving av møtereferat + lese tidligere bacheloroppgaver	1:00:00
Skisser til GIT/mockup + funksjonalitet-rettigheter	3:30:00
Spørreundersøkelse, personvern	1:38:02
● Rapport	48:22:21
Innledning	2:48:37
Rapport	5:26:15
Rapportoppsett	1:37:11
Without description	38:30:18
● Research	50:42:13
Arkitektur	3:50:22

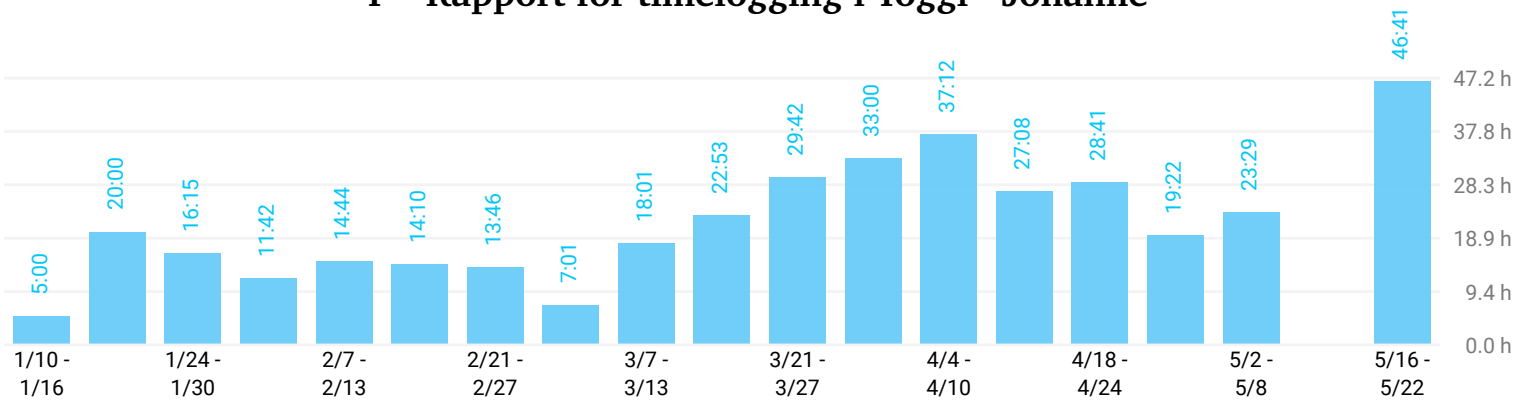
PROJECT - TIME ENTRY	DURATION
Domenemodell	9:54:18
JavaScript	6:59:50
polymer PWA	4:56:19
Polymer PWA	1:21:02
programvaredesign	2:59:39
PWA starter kit opplæring	0:37:37
Research arkitektur	3:05:33
Research arkitektur, teknologi	2:55:00
Research teknologi	6:37:07
Reserch arkitektur	1:34:38
Spørreundersøkelse	5:50:48

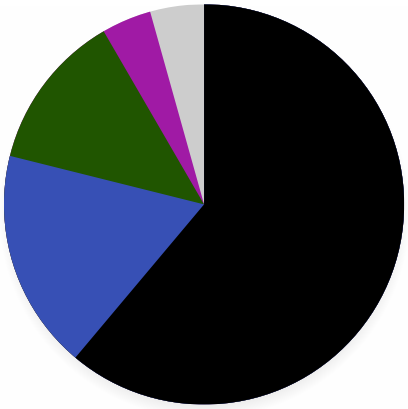
Summary Report

January 10, 2019 – May 19, 2019

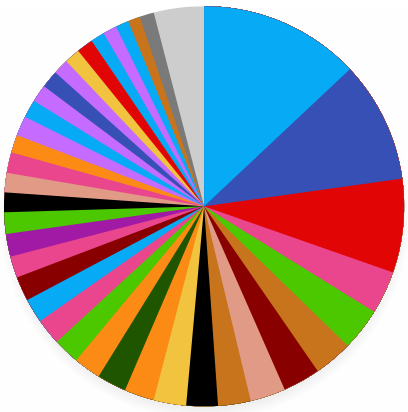
TOTAL HOURS: 388:50:36

F Rapport for timelogging i Toggl - Johanne





PROJECT	DURATION
Koding	237:40:32
Rapport	69:07:38
Prosjektplanlegging	49:27:26
Møter	15:35:00
Other projects	17:00:00



TIME ENTRY	DURATION
Rapportskriving	50:22:24
Arbeidet med prosjektplanen	38:27:26
Oppsett av utviklingsmiljø	29:02:00
Firebase, service workers, push notifications	13:30:48
Server-side scripting, REST API	13:24:42
Tabell for helsepersonell	12:22:00
Arbeidet med rapporten	11:31:00
Forberedelse til utvikling	11:08:00
Tabell for helsepersonell: Endre status for en utøver/finpuss	10:21:11
Mine skader, redigere og arkivere	10:03:16
Grunnleggende innlogging (demo)	10:00:00
Registrering av skade	9:28:00
Express server-klient kommunikasjon	9:04:03
Møte med Terje	8:35:00
Tofaktor-autentisering	8:30:00
Mine skader, tabell for helsepersonell	8:05:27

Brukerregistrering	7:45:00
Vurdering av personvernkonsekvenser	7:14:14
Server-side tilkobling av database, research REST API	7:07:00
Arbeidet med spørreundersøkelsen	7:00:00
Fetch med Redux	6:42:08
Design/koding av Min side	6:30:00
Grunnleggende innlogging	6:01:26
Sikret MariaDB	6:00:00
UI-forbedringer	6:00:00
Research sikkerhet og implementering	6:00:00
Oppsett av SonarQube	5:45:00
Bugfixing, rydding og kommentering	5:30:00
Sikkerhet rundt innlogging og registrering	5:22:39
Databasetilkobling og gyldig HTTPS	5:00:00
Arbeidet med UML-diagrammer	5:00:00
Spillerside	4:48:48
Skaderegistrering og tidssone i MySQL	4:38:22
Innlogging med serverkommunikasjon	4:30:00
Webdesign og optimering	4:00:00
Møte med Tom	4:00:00
Without description	4:00:00
Other time entries	16:00:42

PROJECT - TIME ENTRY	DURATION
Koding	237:40:32
Brukerregistrering	7:45:00
Brukerregistrering og optimering	2:50:42
Bugfixing, rydding og kommentering	5:30:00
Databasetilkobling og gyldig HTTPS	5:00:00
Design/koding av Min side	6:30:00
Express server-klient kommunikasjon	9:04:03

PROJECT - TIME ENTRY	DURATION
Fetch med Redux	6:42:08
Firebase, service workers, push notifications	13:30:48
Forberedelse til utvikling	11:08:00
Grunnleggende innlogging	6:01:26
Grunnleggende innlogging (demo)	10:00:00
Innlogging med serverkommunikasjon	4:30:00
Mine skader, redigere og arkivere	10:03:16
Mine skader, tabell for helsepersonell	8:05:27
Oppsett av HTTPS	1:00:00
Oppsett av SonarQube	5:45:00
Oppsett av utviklingsmiljø	29:02:00
Registrering av skade	9:28:00
Server-side scripting, research REST	3:10:00
Server-side scripting, REST API	13:24:42
Server-side tilkobling av database, research REST API	7:07:00
Sikkerhet rundt innlogging og registrering	5:22:39
Sikret MariaDB	6:00:00

PROJECT - TIME ENTRY	DURATION
Skaderegistrering og tidssone i MySQL	4:38:22
Spillerside	4:48:48
Tabell for helsepersonell	12:22:00
Tabell for helsepersonell: Endre status for en utøver/finpuss	10:21:11
Tofaktor-autentisering	8:30:00
UI-forbedringer	6:00:00
Webdesign og optimering	4:00:00
● Møter	15:35:00
Møte med Terje	8:35:00
Møte med Terje og Dag	3:00:00
Møte med Tom	4:00:00
● Prosjektplanlegging	49:27:26
Arbeidet med prosjektplanen	38:27:26
Arbeidet med UML-diagrammer	5:00:00
Begynt å definere roller og funksjonalitet, litt mockup av funksjonalitet	3:30:00
Diskutert utviklingsmodeller	1:00:00
Laget stikkord til prosjektplanen	0:30:00

PROJECT - TIME ENTRY	DURATION
Research utviklingsmodeller	1:00:00
Rapport	69:07:38
Arbeidet med rapporten	11:31:00
Rapportskriving	50:22:24
Vurdering av personvernkonsekvenser	7:14:14
Research	13:00:00
Arbeidet med spørreundersøkelsen	7:00:00
Research sikkerhet og implementering	6:00:00
Without project	4:00:00
Without description	4:00:00

Spillerstatus

NTNU
Kunnskap for en bedre verden