Vejleder: Christian Budtz | Diplomingeniørprojekt | Afleveringsdato 26/06/2020

Applikation til optimering af møder og undervisning

S161791

Troels Lund

SPINOFF
GØR DIG BEDRE TIL DET DU GØR

Danmarks Tekniske Universitet
DTU

Indholdsfortegnelse

Prøv PWA'en

Brugere:

vadmin@spinoff.nu Spinoff1234
Facilitator@spinoff.nu Spinoff1234
vadmin@spinoff.nu Spinoff1234

Det er muligt at oprette tilknyttet en eksisterende virksomhed.

Virksomheder i demoen:

Virksomheds navn	ID
Spinoff	1
Test Aps	2

Eller oprette en ny virksomhed.

Link til reposetories findes i Bilag 4 - Link til repositories og link til både backend og frontend findes i Bilag 5 - Deployed test version

Abstract

The company Spinoff has provided the assignment for this paper. Spinoff is a consulting firm that helps their customers with organizational-, leadership-, employee- and team development.

With this in mind Spinoff wants to provide a tool for their customers to use. This software should be a tool to evaluate their own performance regarding public speaking and help Spinoff consultants provide better advice and guidance regarding these activities. This could for example be a talk. The idea is that the spectators can provide feedback to the speaker right after the event has finished.

This project aims to provide Spinoff with a simple prototype of this system, which can help them evaluate the system commercially and functionally.

The system is built as a full stack system with a backend part consisting of a .NET Core REST API and a frontend part built in Next JS (React). The frontend is utilizing the progressive web application technology to provide the end user with the feeling of a native mobile application.

The focus of the development has been on the main use cases of the system to provide a decent experience for the user. The system is managing events in order to receive feedback, obtain the feedback and show the feedback to the speaker in a way that is concise and easy to comprehend for the user.

The system further builds on top of this by providing a login system with different user privileges, a very simple user administration module, real time feedback and a simple dashboard to view feedback for a given time period.

To make the system easy to manage, techniques associated with CI/CD have been used to make It easy and effortless to deploy the system. The system has been deployed to the platform Azure and has been used for testing a production version of the system for most of the development.

In conclusion the system has reached a desirable state with all the key features implemented. It is my assessment that the system with little to no further development is ready for testing with a larger user-group.

Though there is room for improvement in both the code base and the user interface. The system should undergo further testing to reach a state, where the product can be released to the public.

Indledning

Projektstiller er Spinoff, der er en konsulentvirksomhed, som tilbyder forløb og konsulentarbejde inden for organisations-, leder-, medarbejder- og teamudvikling[1].

Spinoff ønsker et IT system, som kan underbygge deres allerede eksisterende konsulentforretning ved at give kunderne mulighed for at opsamle data om begivenheder såsom møde, undervisning og foredrag, hvor kommunikation og klarhed er vigtigt for, at alle deltagere har muligheden for at få det optimale udbytte.

Formålet med feedback-systemet er i første omgang at hjælpe Spinoff med at optimere udbyttet deres møder, undervisning eller lignende aktiviteter ved hjælp af feedback givet på stedet kort tid efter begivenheden og dermed give mødelederen/underviseren viden om deres performance ganske kort tid efter de har performet.

Dataene skal derudover kunne akkumuleres for virksomheden eller organisationen, så det også kan danne grundlag for strategiske indsatser og udviklingsområder.

Virksomheden ønsker således også at sælge systemet som et værktøj, der kan bruges af kunder, som en del af Spinoffs service.

Projektstyring og planlægning

Projektet startede med en række møder med projektstiller-virksomheden Spinoff, for at finde frem til

- Stakeholders
- Ønsker og krav til systemet
- Forventningsafstemning

Ud fra disse møder blev der fundet frem til et Project Scope Statement, som kan ses under Bilag 15 - Project Scope Statement, samt formuleret en kravspecifikation og grundlæggende brugsscenarier.

Ud fra disse kunne en grov tidsestimering udarbejdes af projektet[2], som affødte en række ændringer i kravspecifikationen, så den endte med at være, som det kan ses i Bilag 11 - Oprindelig Kravspecifikation. Hvoraf en MVP for projektet blev betragtet som værende “must have” kravene.

Med udgangspunkt i tidsestimatet kunne en foreløbig milestoneplan (herefter benævnt som planen) udarbejdes. Planen sætter nogle delmål/milestones gennem projektperioden.

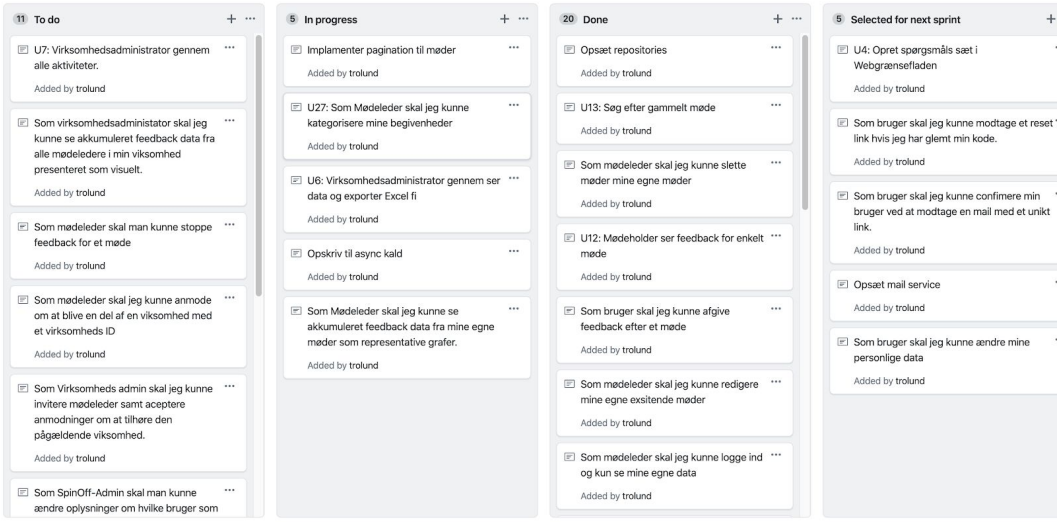
[1] (“Spinoff” n.d.)
[2] Bilag 13

I Bilag 14 - Milestoneplan findes en udgave af milestoneplanen, som den så ud 3 uger før aflevering af denne opgave.

Projektet er blevet udviklet agilt og efter Unified process[3]. Planen er blevet revideret ved hver milestone, og der været scope ændringer, de mest markante af disse kan findes under afsnittet Scope Justeringer.

Jeg har valgt at have en milestone per 4. uge. Desuden er alt arbejde delt op i sprint, som det kendes fra scrum, og hver af disse sprint er på 2 uger og der indgår dermed 2 sprint per milestone. Status for hver sprint har været overleveret på møder med henholdsvis projektstiller samt vejleder.

Under projektet er et Kanban Board desuden brugt til at holde overblik på opgaverne, samt udvælge opgaver til den kommende sprint.



Tidsestimering

Opgaven blev brudt ned til mindre delopgaver, og estimeres på baggrund af erfaringer med lignende opgaver.

Første grove estimat var omkring 160 timer for udarbejdelsen af MVP'en. Dette estimat var omkring det halve af hvad det burde have været.

I retrospekt kan det ses, at der mangler at blive taget højde for en række væsentlige poster, såsom møder og kommunikation med både projektstiller samt vejleder. Og i høj grad også deployment af systemet.

[3] (Larman 2002), side 54, 5.2

Dertil kommer også, at der senere måtte foretages en omlægning af projektet, som kostede meget tid, denne omlægning var der selvfølgelig ikke taget højde for i estimatet.

Hvis disse komponenter havde været en del af grov estimatet havde det højst sandsynligt passet væsentligt bedre.

Projekt Scope

Gennem hele projektet er der løbende blevet justeret på scopet. Cirka 5 uger inde i projektet måtte projektet omlægges. Dette skyldes i høj grad at de valgte teknologier ikke gjorde det muligt at komme i mål med den aftalte MVP. Der blev desværre spildt meget tid på denne omlægning af projektet og gjorde at store dele af backenden måtte redesignes og frontenden måtte fuldstændigt erstattes.

Kunden havde oprindeligt et ønske om en native app til IOS og Android applikation. For at kunne nå det, startede jeg med et React Native(reference) projekt, men undervejs nåede vi frem til, at en PWA dækkede behovet og har en række fordele. Herunder gav en PWA nemmere distribution, hurtigere opdateringer, større delt kodebase med webgrænsefladen og lidt lavere udviklingsomkostninger. Brugsscenarierne forblev uændrede efter omlægningen.

Scope justeringer

Kunden har løbende ønsket yderligere features, som kunne tilføjes til løsningen, hvoraf de som bidrog til kernebrugsscenarierne, er blevet taget ind og resten udskudt til videreudvikling af systemet.

Undervejs blev to mindre justering gjort i forhold til de oprindelige specifikation, disse to var:

- Realtids feedback
- Mulighed for at styre spørgsmåls rækkefølge i et spørgsmåls sæt.

Realtids feedback blev taget ind i projektet af to årsager. En kunde som skulle teste produktet ønskede featuren, før kunden ville teste det og udvikleren på projektet synes det var en god opgave at inkludere.

Muligheden for at styre spørgsmålenes rækkefølge i et sæt, var en overset detalje fra start, da det ikke var blevet tydeliggjort gennem møderne mellem udvikler og projektstiller, at dette var et krav. Derfor blev denne feature indstillet til den kommende sprint umiddelbart efter, at det blev opdaget.

Der er egentlig tale om Scope Creep, men jeg valgte at tage ændringerne med da Spinoff mente at rækkefølgen var vigtig for brugerne.

Analyse

Eksisterende løsninger

Der findes mange eksisterende løsninger hvis formål er meget lignende det system som Spinoff ønsker. Det har dog ikke været muligt for Spinoff at finde et system som har være tilfredsstillende for den.

De eksisterende løsninger har en række problemer i forhold til Spinoff's ønsker.

Mange af systemerne er typisk meget komplekse, hvilket gør dem sværere at bruge, for både spinoff og deres kunder.

Spinoff gerne have et system som samler deres kunder i et system, således at de ikke skal have konti på flere platforme, hvor de måske kun har begrænset adgang og derfor svære ved at vejlede deres kunder.

For flere af Spinoffs kunder er det kritisk at løsningen er på dansk, hvilket også langt fra er tilfældet med mange af eksisterende løsninger.

Som en del af filosofien bag systemet som Spinoff ønsker, er at de gerne vil have deres kunder til at give feedback meget ofte, men stille dem færre spørgsmål og et meget enkelt svar skala, så det bliver overskueligt at svare lige efter eksempelvis et mødet. En platform som lægger op til dette, er heller ikke blevet fundet.

På nuværende tidspunkt bliver feedback skrevet ned på små posted notes og skrev ind i et Excel-ark senere hvilket er en langsomlig manuel process, og gør det samtidigt svært at sammenligne på tværs tid hvor flere ark skal kombineres.

I bund og grund ønsker de at kombinere en tjeneste som Kahoot, der giver nem mobil-venlig adgang og spørgeskema tjeneste som SurveyMonkey. Hvor de kan så kan forene spørgeundersøgelser med standardiserede spørgsmål, som kan give overblik over udviklingen over tid på en simpel måde.

Risikoanalyse

Da dette projekt bliver drevet af blot en person, vil sygdom være en væsentlig faktor, som vil kunne stoppe fremdriften af projektet delvist eller fuldkomment. Derfor er denne faktor vurderet meget højt. Som det kan ses af Tabel 1 er desuden vurderet en høj risiko for fejl i tidsestimering. Dette er gjort da projektet er udviklet med teknologier som jeg ikke havde indgående kendskab til, det blev det vurderet som meget sandsynligt at estimering, ikke ville passe nøjagtigt.

Identificeret risici

Tabel 1 beskriver nogle af de mest sandsynlige risici eller risici med højt påvirkning af projektet.

Id	Risk	Impact (0-5)	Probability (0-5)	Score (impact * probability)
R1	Sygdom (Langvarigt)	4	2	8
R2	Sygdom (kortvarigt)	1	4	4
R3	Forkert teknologi valg	4	2	8
R4	Tidsestimering er underestimeret.	3	4	16
R5	Mistet kode	3	2	6
R6	Arbejde på ikke relevante features	2	3	6
R7	Fejl i brugte biblioteker	2	1	2
R8	Server down time	2	1	2
R9	Begrænsninger på Azure service	2	2	4
R10	Spinoff dropper projektet	5	1	5
R11	Spinoff dropper samarbejdet	5	2	10
R12	Mistet dokumentation	4	2	8
R13	Uidentificeret bugs	3	3	9
R14	Centrale brugsscenarier mangler at blive identificeret	5	1	5
R15	Perifere brugsscenarier mangler at blive identificeret	2	2	4
R16	Datamodel er mangelfuld	2	2	4

Tabel 1: Mest relevante risici

Matrix over risici

Probability Impact	1 <10%	2 10-25%	3 25-50%	4 50-75%	5 >75%
5 (Fatal)	R10, R14	R11			
4		R1, R3, R12			
3		R5	R13	R4	
2	R7	R9, R15, R16	R6		
1 (Minimal)		R8		R2	

For at Imødegå risici er det lavet planlagt strategi for alle risici som er mellem- eller høj-risikozonen.

Risiko	Score	Strategi
R1	8	Holde sig isoleret og passe på sig selv.
R3	8	Løbende evaluere fremgang og identificere hvad der hindre fremdrift i projektet.
R12	8	Lave backup af dokumentationen løbende
R13	9	Forsøge at teste produktet så meget som muligt, gerne med externe testere.
R6	6	På minde ufokuseret udvikler hvad de primære brugsenariere indebære.
R10	5	Hold Spinoff orienteret og løbende forventningsafstemme løsningen.
R14	5	Undersøg brugsmønstre grundigt i samarbejde med Spinoff og slutbrugere.
R11	10	Hold Spinoff orienteret og løbende forventningsafstem løsningen.
R4	16	Løbende evaluering af fremgang og tidsforbrug samt justere hvis nødvendigt.

Da der kun findes en person på projektet som både må agere udvikler og projektmanager, er det denne persons ansvar at imødekomme disse risici med de planlagte strategier.

Brugsscenarier

Sammen med Spinoff identificerede jeg fire slags aktører, hvoraf tre er registrerede brugere i systemet med forskellige roller.

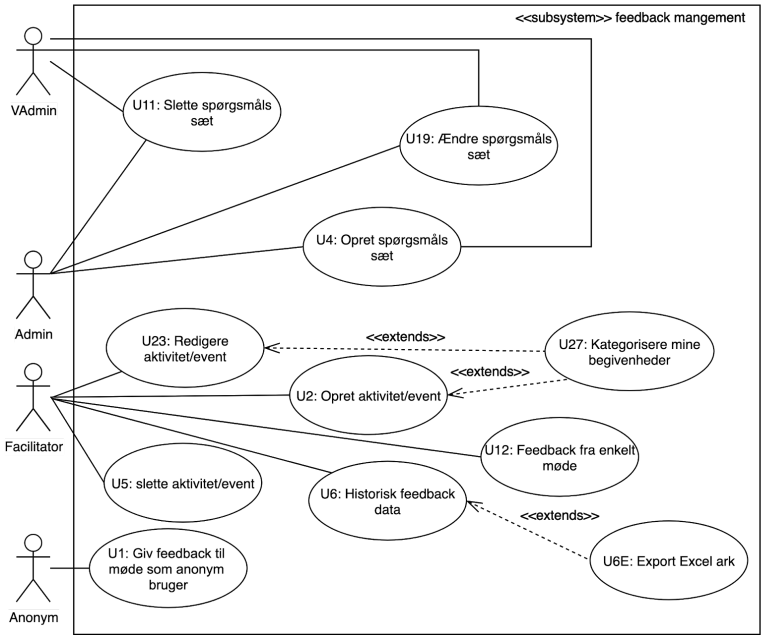
De fire aktører er som følger:

- Registrede brugere
    - Virksomhedsadministrator (VAdmin)
    - Facilitator
    - Administrator (Admin)
- Ikke registreret brugere
    - Anonym bruger

Brugsscenarie diagrammerne figur 1.1 og 1.2, viser de centrale brugsscenarier og hvordan de forskellige aktører i systemet interagerer med systemet. Der er vist to subsystemer, det éne subsystem beskæftiger sig primært med brugeradministration, og det andet primært med feedback.

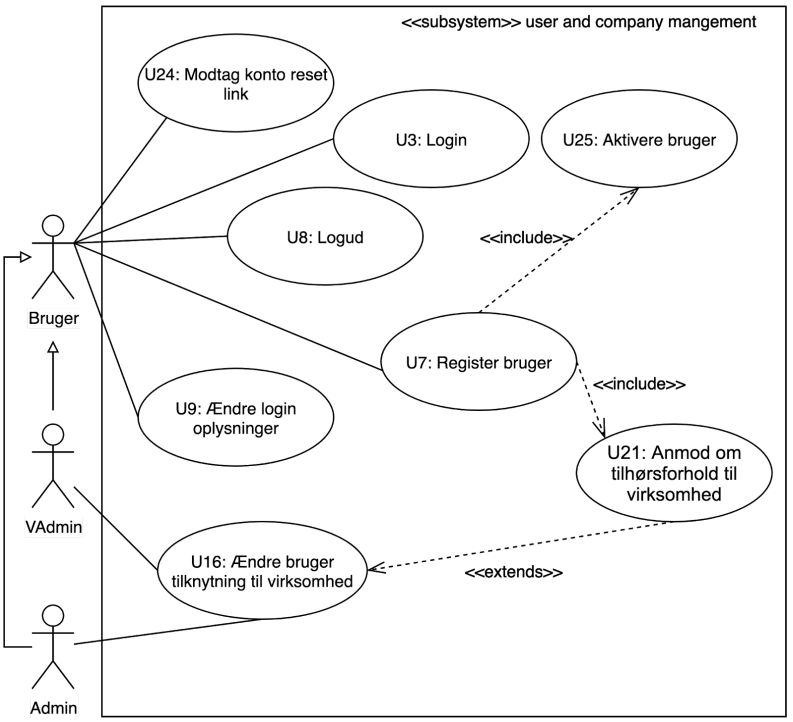
De primære brugsscenarier består af U1, U2 og U12. Disse tre brugsscenarier udgør kernefunktionaliteten som systemet implementere.

- U1 - Som anonym bruger skal jeg kunne afgive feedback efter et møde/foredrag/undervisning.
- U2 - Som facilitator skal jeg kunne oprette en aktivitet/event.
- U12 - Som facilitator skal jeg kunne se feedback for et enkelt event



Figur 1.1 - Use case diagram (feedback system)

Admin og VAdmin har desuden ret til alt hvad Facilitator gør, pilene er ikke tegnet ind, da diagrammet ville blive uoverskueligt.



Figur 1.2 - Use case diagram (Bruger og virksomhedsadministration system)

En oversigt over alle brugsscenarier der er udarbejdet findes i Bilag 1 - Oversigt over brugsscenarier, derudover findes der en traceability matrix mellem krav og brugsscenarier i Bilag 2 - Brugesscenarier og krav traceability matrix og beskrivelserne findes i Bilag 18 - Brugsscenarie beskrivelser.

Kravspecifikation

Kravspecifikation er lavet som en kombination af FURPS[4] og MoSCoW[5]. Kravspecifikationen er udarbejdet i samarbejde med kunden og er derfor udtryk for, at både tekniske og kommercielle overvejelser i prioriteringen.

Functionality

Must

1. Systemet skal kunne oprette møder
2. Systemet skal kunne slette møder
3. Systemet skal kunne rette informationer om et eksisterende møde.
4. Systemet skal indeholde et login system med 3 brugerroller
    a. Mødeleder/Underviser (Fasilitator)
    b. Virksomhedsadministrator (VAdmin)
    c. Spinoff-administrator (Admin)
5. Systemet skal kunne gemme historiske feedback data for afholdte møder
6. Systemet skal være i stand til at visualisere akkumuleret historisk data.
7. Systemet skal kunne vise alle oprettede aktiviteter for en given burger.
8. Systemet skal kunne vise en Virksomhedsadministrator alle aktiviteter i en virksomhed.
9. Systemet skal gøre det muligt for en bruger at tilhøre en virksomhed.
10. Systemet skal gøre brugeren i stand til at ændre sine egne login oplysninger.
11. Systemet skal kunne vise møderne i en Gregorian calendar.
12. Systemet skal kunne holde personlige data på en sikker både og i henhold til GDPR.
13. Systemet skal kunne vise en GDPR (Vilkår for tjenesten) besked ved oprettelse af en bruger.
14. Systemet skal kunne visualisere feedback for et enkelt møde/event.
15. Systemet skal kunne søge efter møde på navn, beskrivelse og kategori.
16. Systemet skal kunne vise alle kommende møder og afholdte møder som en liste.
17. Systemet skal kunne generere strategisk rapporter som Excel filer fra det historiske data.
18. Systemet skal definere nye spørgsmåls-sæt såfremt virksomheden er betalende bruger.
    a. Herunder bør det være muligt at lave spørgsmåls-sæt med uvilkårlig antal spørgsmål.
19. Systemet burde være i stand til at validere feedback data således at kun “korrekt” data ender i systemet.

[4] (Larman 2002), side 56, 5.4
[5] (“Chapter 10: MoSCoW Prioritisation” n.d.)

Should

1. Systemet burde kunne filtrere visning af historisk data på mindst et niveau, niveauer kunne være:
    a. Hvilke type/emne mødet har
2. Systemet burde kunne automatisk stoppe feedback for et møde efter et tidsinterval på 12 timer efter slut tidspunktet.
3. Systemet burde have mulighed for brug af QR Koder til nemt at kunne give feedback på givent møde.
4. Systemet burde have mulighed for at mødeledere live kan se deres feedback efterhånden som det bliver givet, herunder også antal deltagere, som har besvaret.
5. Systemet burde være i stand til at sende mails ved konfirmering af kontooprettelse.

Could

1. Systemet kunne have mulighed for at tilpasse feedback-tidsintervallet (hvor lang tid efter deltagere kan give feedback efter mødet)
2. Systemet kunne have mulighed for at angive antal deltagere for et møde så det kan markeres som at have fuldendt feedback.
3. Systemet kunne være i stand til at sende mails med glemt kodeord.

Won't

1. Systemet vil ikke kunne sende mails til alle mødedeltagere.
2. Systemet vil ikke være istand til at indeholde et invitationssystem/administrationssystem således at Virksomhedsadministrator kan acceptere eller afvise mødeledere for den givne virksomhed.
3. Systemet vil ikke have automatisk notifikation til mødeleder.'
4. Systemet kunne have automatisk notifikation / reminder til deltagere der mangler at besvare.
5. Systemet kunne gøre det muligt at definere lokationer så som "mødelokale 2, bygning 2"
6. Systemet kan have en "White label" løsning som lader kunder customize brugergrænsefladen til deres egen design guidelines. Her under:
    a. Logo
    b. Farver
    c. Typografi
    d. Evt. special login på subdomæne.
7. Systemet kunne gøre Virksomhedsadministration i stand til at definere grupper inden for organisationen, fx. "Marketingafdelingen", "Salg" eller "IT".
8. Systemet kunne være i stand til at "pushe" aktiviteter til brugerens online kalender, samt modtage aktiviteter fra samme online kalender. Dette kunne være (I prioriteret rækkefølge):
    a. Microsoft Outlook calendar
    b. Google calendar

Usability

1. Systemet skal kunne betjenes af ikke tekniske personale med lille til ingen support.
2. Systemet skal gøre det nemt for en deltagere at afgive feedback dvs. Under 3 tryk/klik for at kunne svare på spørgsmål.
3. Systemet burde kunne vise flotte animationer som underbygger hvad bruger ønsker.

Reliability

1. Systemet skal kunne håndtere conference størrelse store forsamlinger (2000-3000 deltagere) stabil vis.
2. Systems nedetid skal være mindre end 2%.
3. Systemets møde opslag skal være sikkert nok til at det under 1/18000 gange er det korrekte møde som findes.

Performance

1. Systemet skal virke hurtige og responsivitet for bruger det vil sige der ikke må opleves længere ventetider end max 0.5sek.
2. Systemet skal kunne være skalerbart til at kunne håndtere +1 million møder.

Supportability

1. Systemet burde kunne køre med minimal support fra Spinoff medarbejdere.
2. Systemet skal være dokumenteret i en grad så det er muligt for en udefrakommende udvikler kan videreudvikle på projektet efter denne er læst såfremt udvikleren allerede har kendskab til de brugte teknologier.

Design constraints

ingen

Implementation constraints

1. Udviklingen samt dokumentation skal kunne foregå indenfor den aftalte projekt tid.

Interface constraints

1. Produktet bør kunne bruges både på mobile enheder såvel som en PC, men bør fokusere på mobile platforme.

Physical constraints

ingen

Domænemodel

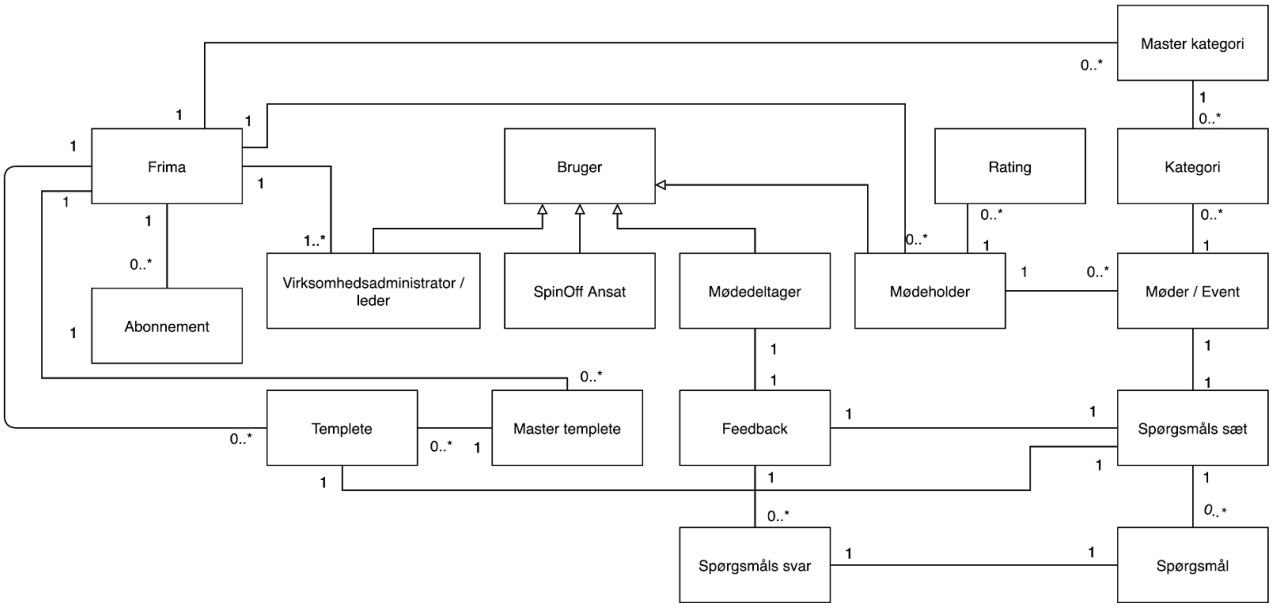
Som en del af analysen og opdagelsen af domænet og en evt. sætningsstruktur blev en domænemodel udarbejdet.

Som det kan ses af figur 2 kan brugeren være de fire forskellige typer som før beskrevet. Mødeholdere (facilitatoreren) og virksomhedsadministrator tilknyttet en virksomhed, der skal være mindst en virksomhedsadministrator tilknyttet en virksomhed.

Mødeholderen knytter sig til sine egne møder, og disse møder kan være associeret med en række kategorier. Mødet er ligeledes bundet op på et specifikt spørgsmåls sæt. Mødeholderen skal have en rating, som skal afspejle han/hendes overordnet performance.

Modelleringen af tilbagemeldingerne (feedback) er tænkt, således at de indeholder en række svar på spørgsmål, som findes i et spørgsmålssæt. Tilbagemeldingerne har en direkte tilknytning med et bestemt spørgsmålssæt, og et svar er tilknyttet til et specifikt spørgsmål i det sæt.

Et spørgsmåls sæt er associeret med en template, som er associeret med en Master template administreret af et firma, samme mønster gør sig gældende for kategorier.



Figur 2 - Domænemodel

En større udgave af domænemodellen finde i Bilag 8 - Domæne model.

Der er ikke blevet implementeret en betalingsløsning, men som det kan ses, planlægges det, at et abonnement vil knytte sig til en virksomhed.

Valg af teknologier

Teknologierne er valgt ud fra, at de er alment benyttet og forventes at blive videreudviklet mange år frem. Dette gør også at det typisk vil være relativt let at finde udviklere med erfaringer inden for den givne teknologi, såfremt kunden ønsker at videreudvikle produktet.

Derudover har det også været vigtigt for mig at vælge teknologier, som ser ud til ikke foreløbig at aftage væsentligt i anvendelse, for at sikre at de anvendte teknologier ikke bliver forældede tidligt i produktets livscyklus.

Research

En del research blev foretaget ved forprojektet, men denne reseach var primært omkring native-hybrid løsninger, såsom Flutter og React Native.

Da det foregående projekt ikke så ud til at leve op til forventningerne, blev der endnu engang gennemført en research fase denne gang primært centreret omkring web teknologierne og Progressive web application (PWA).

Efter at have valideret, at PWA teknologien var istand til at opfylde kravene til den mobile brugeroplevelse, som Spinoff ønskede, blev researchen primært rettet mod, hvilke frameworks som kunne anvendes til PWA'er.

Frameworks som blev undersøgt:

- Angular
- React
- Vue
- Svelte
- NextJS (React)

Angular, React og Vue kan overordnet set betragtes som frameworks. Svelte er en relativt ny teknologi som reelt set er en compiler, men muliggør mange af de samme features som frameworkene gør. Svelte er en meget interessant teknologi, som producerer sider som fylder mindre end framework'sene, hvilket i sidste ende vil give brugeren oplevelsen af en mere responsiv side. Svelte er dog meget nyt, hvilket betyder, at der er langt mindre information og hjælp at hente online sammenlignet med de andre omtalte teknologier, hvilket gjorde denne teknologi blev valgt fra.

Angular har en tendens til at producere væsentligt tungere (større) produktions applikationer på grund af en øget mængde kode fra selve frameworket, i det endelige produkt[6]. Dette

[6] (Sethi 2018)

virker uhensigtsmæssigt i en prototype, som skal være et relativt lille projekt og ikke mindst loade og køre hurtigt på mobile enheder.

Vue er et interessant framework som jeg personlig godt kan lide. Vue producerer de mindste kode bundles til produktions applikationen (Af frameworks'ene), Vue blev valgt fra fordi React trods alt stadig har en væsentlig større adoption og brugerbase hvilket giver en række positive effekter.

Valget faldet på React, som nu i en årrække været et af de mest brugte Frontend Javascript framework, og i høj grad er den dominerende af tre frameworks[7].

Derudover vil Typescript blive anvendt, dette har jeg valgt da jeg har god erfaring med det fra andre sammenhænge. Typescript er et statisk compiled sprog om transpiler til javascript. Typescript gør Javascript type-stærkt, hvilket kan være en kæmpe fordel ved udvikling af større systemer[8], da linteren kan give fejlmeddelelser ved mere end blot syntaktiske fejl.

React er som udgangspunkt et client-side framework, men jeg har valgt at bruge en "overbygning" kaldt Next JS som giver muligheden for SSR (Server side rendering) samt SSG (Static Site Generator)[9]. Derudover optimerer NextJS også kode ved hjælp af code splitting. Det vil sige, at den laver "bundles" af koden til hver enkelt side[10]. Det smarte ligger så i, at den også laver bider af kode, som deres egne bundles, der bliver brugt mere generelt. Dette gør, at kode som bliver brugt på tværs af sider kun behøver at blive hentet en gang, og det vil i sidste ende få brugeren til at opleve, at applikationen føles hurtigere.

Som alle andre frameworks kommer Next Js med en række konventioner for, hvordan tingene skal gøres. Dette betyder, at man i mange tilfælde bliver tvunget til at gøre det på en specifik måde. Dette har sine fordele i og med, at man laver tingene med en velafprøvet metode, men kan også gøre løsningen mindre fleksibel.

Next js kommer out-of-the-box konfigureret med Hot Module Replacement, hvilket gør det muligt at udskifte dele af kodebasen under runtime. Dette kan øge produktiviteten væsentligt på langt sigt, da udvikleren får visuel feedback langt hurtigere end ved et refresh af siden ved hver ændring.

Den primære grund til at vælge at bruge Next JS i dette projekt er den forbedret performance (Se Bilag 20 - Lighthouse evaluering), og de specifikke pakker som findes til opsætning af en PWA i NextJS.

Derudover vil SSR også forbedre SEO[11], når Spinoff skal have mere information om selve produktet på siden, som de ønsker, at folk nemt skal kunne finde gennem søgemaskinerne. Jeg havde desuden hørt meget godt om Next js og ønskede at prøve det.

[7] ("Top Front-End Frameworks in 2020 | Existek Blog" 2020)
[8] (Bähr 2017)
[9] ("Learn | Next.js" n.d.)
[10] (Copes 2019)
[11] (Chandu 2019)

Til backenden havde jeg et ønske om at anvende .NET Core platformen, da .NET Core og dets biblioteker indeholder meget af den funktionalitet, som var nødvendigt for projektet. Derfor blev der også valgt en relationel database frem for No-SQL for at kunne anvende en ORM som jeg også ønskede at udforske.

MSSQL blev valgt som database da dens EF Core fungerer godt sammen og at den slags database var nem at sætte op på Azure ved deployment.

Hvorfor PWA?

Den største motivation for at vælge PWA teknologien til dette projekt har været at have en enkelt fælles kodebase mellem platformene, hvilket har gjort det nemmere for mig ene mand at nå i mål med en tilfredsstillende prototype, som ikke vil være afgrænset til en enkelt platform.

Ved native udvikling vil man typisk distribuere applikation via en app store. Denne tilgang kan både være en længerevarende proces, da koden i nogle tilfælde skal kvalitetssikres/trussel undersøges, før den kan findes i app storen, og det kan være dyrt for firmaet bag, da der samtidig skal betales for registrering og gebyre.

Dette problem har PWA'er slet ikke, da de meget simpelt kan lægges op på en server og distribueres den vej igennem. Hvilket igen er factor der gør at jeg langt nemmere har kunne distribuere den på mine egne vilkår og får den testet løbende hos Spinoff. Yderligere har PWA'en en klar fordel ved at den kan bruges i browseren uden installation, hvilket især kan betragtes som en fordel i brugerscenarie U1, hvor brugeren giver feedback. På den måde bliver brugeren ikke tvunget til at hente og installere en app for at give feedback én gang.

Ligeledes er opdateringen af app'en også meget nem, da dette blot er at udskifte filerne som er på serveren med den nye version. Brugerne vil dernæst gradvist få den nye udgave, mens de bruger den.

Service workers

Service workers er i meget høj grad magien bag Progressive Web Applikationer. Service workeren kører på en seperat tråd og har ingen adgang til DOM træet. Derimod har den adgang til nogle features, som før var forbeholdt native applikationer såsom push notifikationer[12].

Service worker implementerer en proxy mellem klienten og serveren, som kan serve cached data, således at der selv når klienten er offline vil app'en kunne vises[13]. Hvilket kan forbedre oplevelsen væsentligt for brugerne, især på mobil enheder hvor forbindelsen kan svinge kraftigt.

[12] ("Service Worker API" n.d.)
[13] ("Making PWAs Work Offline with Service Workers" n.d.)

Installerbar

PWA'er er også installerbare på både Android og IOS, og kan findes på hjemmeskærmen, og browsere som supporterer PWA'er vil have appen i browseren.

PWA'er er i høj grad stadig under udvikling i 2020 og er ikke helt blevet standardiseret på tværs af alle browsere og platforme. Dette betyder også, at der stadig ikke er klare standarder for alt, her i blandt fx. events involveret i installation af selve appen, herunder *appinstalled* eventet[14] eller *BeforeInstallPromptEvent*[15]. Dette besværliggøre udviklingen, da der skal tages højde for browsers implementering uden nogen form for pollyfil til at hjælpe med compatibility.

En uddybende forklaring af PWA teknologien, samt dens fordele og ulemper kan findes i Bilag 23 - Progressive webapplikationer som teknologi.

Design

Dette afsnit vil gennemgå centrale design valg for systemet, samt design af brugergrænsefladen.

Overordnet design

Overordnet set er applikationen opbygget med klient-server arkitektur. Serveren vil udstille et REST API som vil kunne udlevere alle de nødvendige data til klienten. Applikationen er i høj grad datadrevet, idet hele formålet med applikationen er at opsamle data ved en brugers feedback og bagefter præsentere data på en brugervenlig og overskuelig måde.

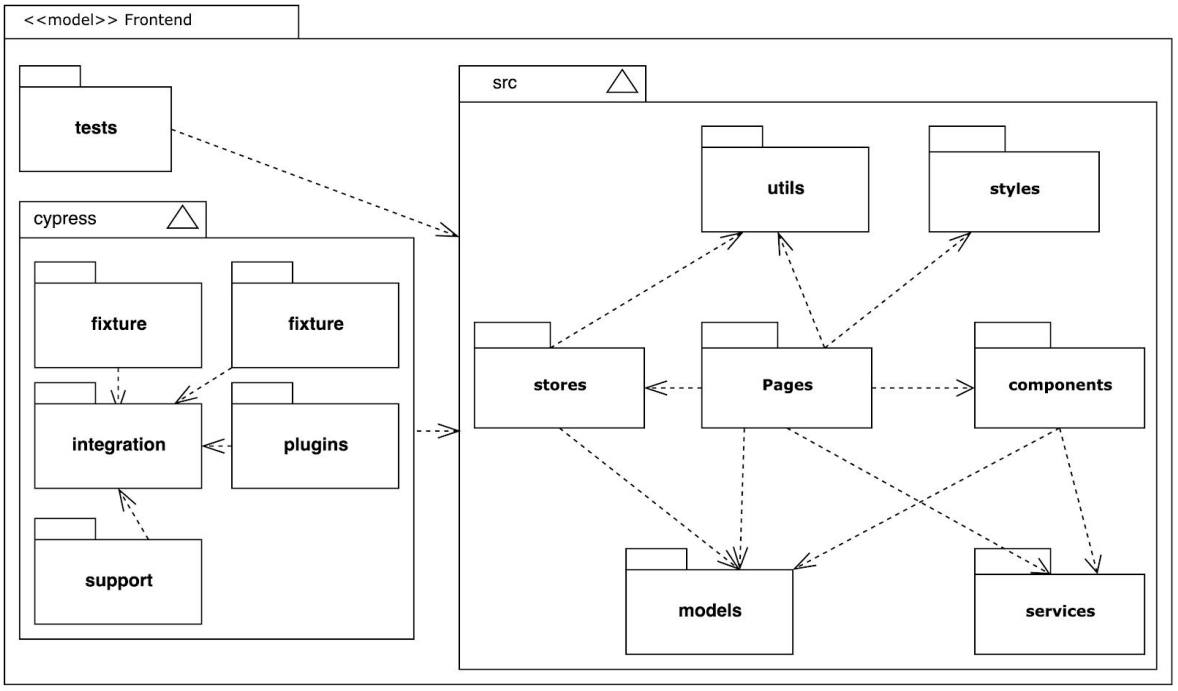
Klient-server arkitekturen er typisk anvendt til sådan type systemer og vil kunne performe godt. Klienter vil naturligt skalere horisontalt, mens serveren både vil kunne skalere vertikalt, ved blot at anvende kraftigere hardware eller skalere horisontalt ved at introducere en load balancer foran, og derved køre flere instanser distribueret ud over evt. flere maskiner.

Frontend

Ligesom så mange andre moderne Frontend frameworks baserer React sig på komponenter. Komponenter giver den åbenlyse fordel, at man kan splitte kode op, således at unødig kompleksitet bliver skjult, hvilket resultere i en mere overskuelig kodebase. Et godt komponentdesign mindsker også udviklingstiden, da godt designet komponenter, kan bruges i andre sammenhænge for på den måde at forbedre reusability.

Derfor er det selvfølgelig forsøgt at designe komponenterne således at de kan bruges i flere sammenhænge.

[14] ("Window: Appinstalled Event" n.d.)
[15] ("BeforeInstallPromptEvent" n.d.)



Figur 3 - Pakke diagram frontend

Sider (pages) er omdrejningspunktet for frontenden, da det er disse, som beskriver det endelige view. Som det kan ses ud fra figur 3, anvender pages således komponenter til lave en grafisk præsentation for brugeren.

“Services" er logik som bliver brugt på tværs af flere sider. “Model” indeholder alle de typer af data som bliver udvekslet mellem frontend og backend, hvor mange af dem vil være identiske med DTO’er i backenden. “Stores” indeholder klasser som beskriver globale variable for applikationen.

“Cypress” er til integrationstest og vil blive beskrevet under Integrations testing. “Tests” er unit tests og vil blive beskrevet i Unit testing.

UX og UI design

Gennem designprocessen er der blevet skitseret wireframes sammen med projektstiller. Der er ikke blevet udarbejdet en UI prototype af designet hvilket ville have været en fordel, men da fokuset af denne opgave ikke har været brugergrænseflade-design og brugeroplevelse er dette blevet nedprioriteret.

Overordnet brugergrænseflade overvejelser

Fra et UX standpunkt er PWA teknologien meget svær at designe UI til. Dette skyldes, at brugerne fra de forskellige platforme er vant til hver deres måde, som brugergrænsefladen er opbygget på. Dette skyldes også i høj grad, hvordan input bliver givet, med henholdsvis fingrene på en touchskærm på en relativ lille skærm sammenlignet med en mus og keyboard på en computerskærm. Disse to oplevelser er meget forskellige og kan hver deres ting.

PWA lægger op til at skulle kunne favne dem begge. I dette projekt har der været fokus på primært mobilplatformen, da dette var kundens ønske. Det skal dog siges at visse features fungerer bedre på henholdsvis mobil eller desktop.

Problemer med forskellige skærmstørrelser kan der tages højde for både ved sideopbygning (DOM'en) og styling. Det kan dog være meget tidskrævende at justere designet til at fungere på alle skærmstørrelser. Det er forsøgt at favne bredt i forhold til forskellige skærmstørrelser, men kan ikke forventes at være fuldt funktionelt på skærme under 4,7".

PWA'en anvender i nogle grad et responsivt design som justerer sig ud fra størrelse på viewporten, såkaldt Responsive Web Design (RWD). Dette bliver beskrevet yderligere i Styling afsnittet.

App vs Web

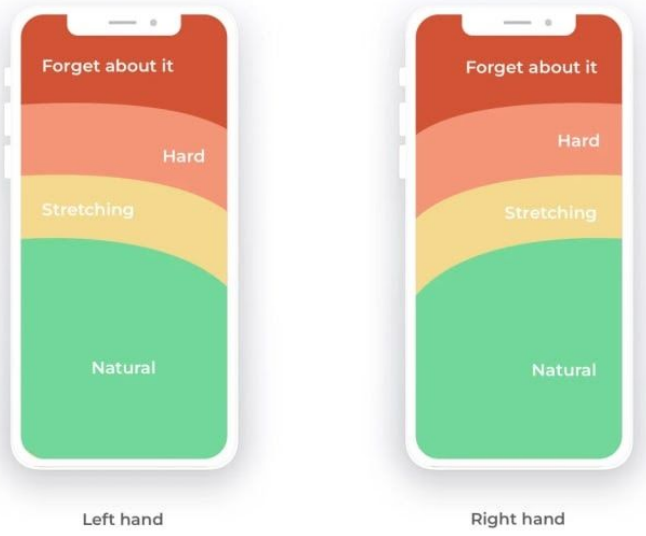
For at give brugeren oplevelsen af en app, er det vigtigt, at app'en også visuelt fremstår, som brugeren forventer, at en app ser ud. I dette projekt er dette gjort ved at imitere UI komponeret og opførsel af en mobilapplikation. Dette betyder eksempelvis, at der i mange tilfælde bliver vist en on-screen tilbage-knap, som man ville gøre det i en native mobilapplikation, men ikke traditionelt på hjemmesider.

Hovednavigation i brugergrænseflade

Mobile enheder har haft en tendens til at blive større samtidig med at størrelsesforholdet mellem siderne henholdsvis bredde og højde også har ændret sig i retning af højere smallere skærme[16].

Dette påvirker selvfølgelige brugeroplevelsen en hel del, da flere og flere har mobilenheder har dimensioner som gør det besværligt at nå toppen af skærmen og derfor kræver to hænder. Som det kan ses på figur 4. Er det faktisk blot den nederste halvdel af skærmen som bruger typisk vil have lettest ved at interagere med.

[16] (Bhagat and Bajaj 2018)



Figur 4 - Enhånds navigations zoner[17]

Dette betyder også at det overordnet vil være en fordel at placere elementer som jævnligt interageres med på den nederste halvdel af skærmen. Noget af det brugeren vil gøre allermest i appen er at navigere mellem forskellige sider.

Derfor er navigationen som anvendes, så vidt muligt "bund navigation", som det tilfældet med hovedet-navigationsbaren som kan ses i figur 5.

Figur 5 - Bund navigation i appen

Dette er ikke noget nyt og har eksisteret i native apps som UI komponenter i både Googles material[18]- og Apples[19] design guides altid.

I dag overgår mobilenders aktivitet på nettet, den traditionelle PC[20], og teknologier som fx. Progressive Web Apps pusher UI design som lægger sig langt tættere op af app UI's end traditionelt web design. Derfor har det været vigtigt for produktet at afspejle den mobile brugergrænseflade.

"bund navigation" har dog også visse ulemper sammenlignet med fx. en navigation drawer[21] /burger menu/side menu, hvor den primære fordel fremfor navigation drawer'en er at den har plads til væsentligt mere indhold da den typisk udfoldet vil fylde størstedelen af skærmen. Det kan dog også ses som en ulempe, at brugeren ikke kan se sine muligheder for

[17] (Felix 2020)
[18] ("Material Design" n.d.)
[19] (Apple Inc n.d.)
[20] ("Mobile Vs. Desktop Internet Usage (Latest 2020 Data)" n.d.)
[21] ("Bottom Navigation - Material Components for Android" n.d.)

navigation uden at have foldet menuen ud, men på den anden side skjules unødige informationer, når de ikke bliver brugt.

For at adressere denne mangel på plads findes der en “mere” knap i bund navigationen som i skrivende stund fungerer ved at navigere til en ny side, som så har de muligheder som mangler. Det kunne til overvejes, om denne i stedet for navigation til en ny side skal fungere mere som en sidemenu, som blot slider ud for på den måde at kombinere de to typer navigation.

Denne ekstra navigation er selvfølgelig frustrerende for brugerne, men først og fremmest vil denne mulighed kun blive vist for et subset af brugerne, nemlig virksomheds-administratorerne og administratorerne. Sider som ligger på denne underside vil desuden være de mindre brugte sider, således at de hyppigst brugte sider nemt kan navigeres til.

Screen Flow Diagram

For at gøre det nemt for brugeren at bruge hjemmesiden uden at have en konto, giver side roden mulighed for at indtaste møde ID eller scanne QR-kode. Dette er visualiseret i Bilag 10 - App flow. Når app'en starter op og brugeren er logget ind vil den automatisk viderestille til oversigts-siden[22]. En kort beskrivelse af centrale skærmbillede kan findes i Bilag 19 - PWA sider.

PWA Installation UX

På Android er det meget nemt at installere en PWA og det kan gøres med en in-browser trigger event. På IOS er det dog en proces af flere trin, og det er ikke muligt at lade brugeren installere ved at klikke på en knap.

Dette gør, at det er nødvendigt at skelne mellem de to platforme, således at brugeren kan gøre det, som er nemmest for den givne platform. I dette projekt er det løst ved, at Android, Chrome-browser samt alle andre browsere, som supporter PWA'er viser en installationsknap, og IOS Safari viser en guide til de steps, som skal foretages for succesfuldt at installere appen.

Backend

Løsningsarkitektur

Løsningen er delt op i fem forskellige projekter samlet i *FeedbackBackend.sln* løsningen.

De fem projekter er som følgende:

- WebApi (REST interface)
- Business
- Data
- Infrastructure
- Test

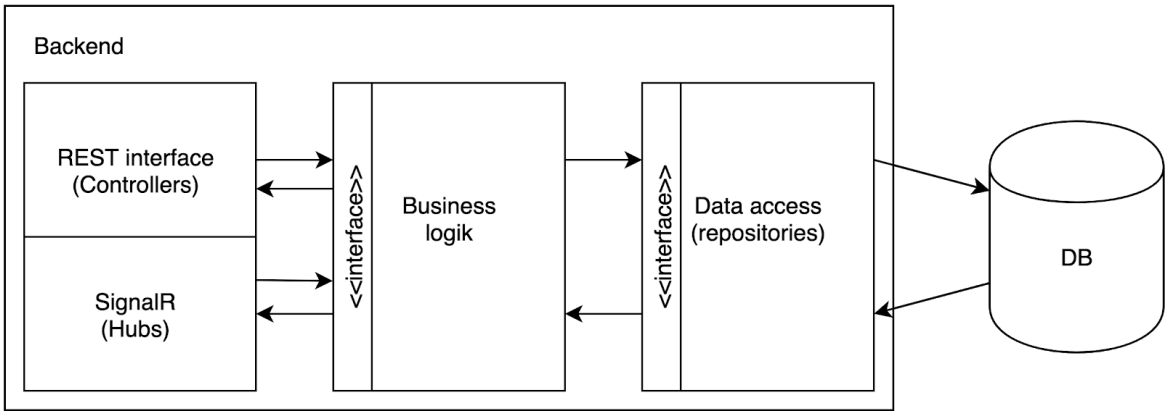
WebApi projektet er et .NET MVC projekt som definerer hele opsætningen.

Test projektet er et XUnit projekt og har referencer til al kode således, at der kan testes på alle niveauer af applikationen. Test projektet er ikke en del af selve backend projektet og bliver derfor heller ikke kompileret ved start af projektet, men skal kompilere og køres separat, som det bliver beskrevet i Unit testing backend afsnittet.

Alle andre projekter er blot klasse biblioteker og kan derfor ikke køres selvstændigt.

[22] Bilag 10.6

Løsningen er designet lagdelt som det kan ses på *figur 6* hvilket betyder at de requests som REST interfacet eller SignalR modtager bliver passet ned gennem lagene for at blive behandlet, og returneret når data er til stede i hukommelsen.



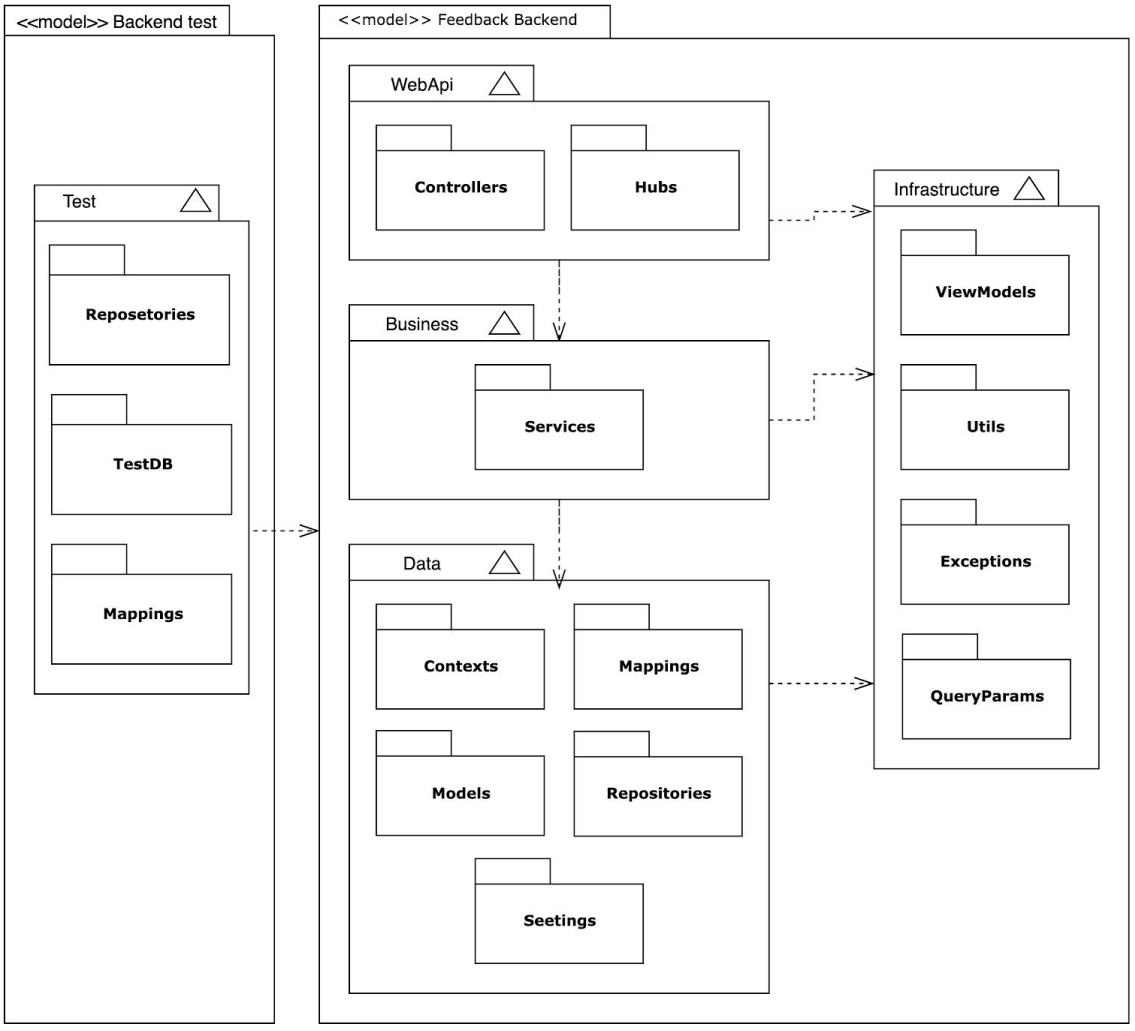
Figur 6 - Løsningsarkitektur afhængigheder, (Dette diagram følger ikke UML standarden)

Applikationen er designet, så den i høj grad benytter sig af dependency injection. Dette giver nem adgang til de registrerede instanser, samt medfører en lav kobling mellem klasserne. Dette øger også testability da selvsamme instanser kan registreres i test projektet.

Afhængigheder og pakke opbygning

Som det kan ses på figur 7, har projekterne afhængighed af laget/projektet under osv. Det vil også sige, at REST interfacet indeholdende controllerne og hubs, der ikke direkte anvender data tilgangs laget, men derimod anvender en service, som indeholder den nødvendige logik for at håndtere dele eller fuldstændig, hvad det givne endpoint skal opnå.

Controllerne dækker over en række klasser som beskriver REST interfacet og Hubs beskriver endpointet som SignalR håndterer.



Figur 7 - Pakke diagram backend

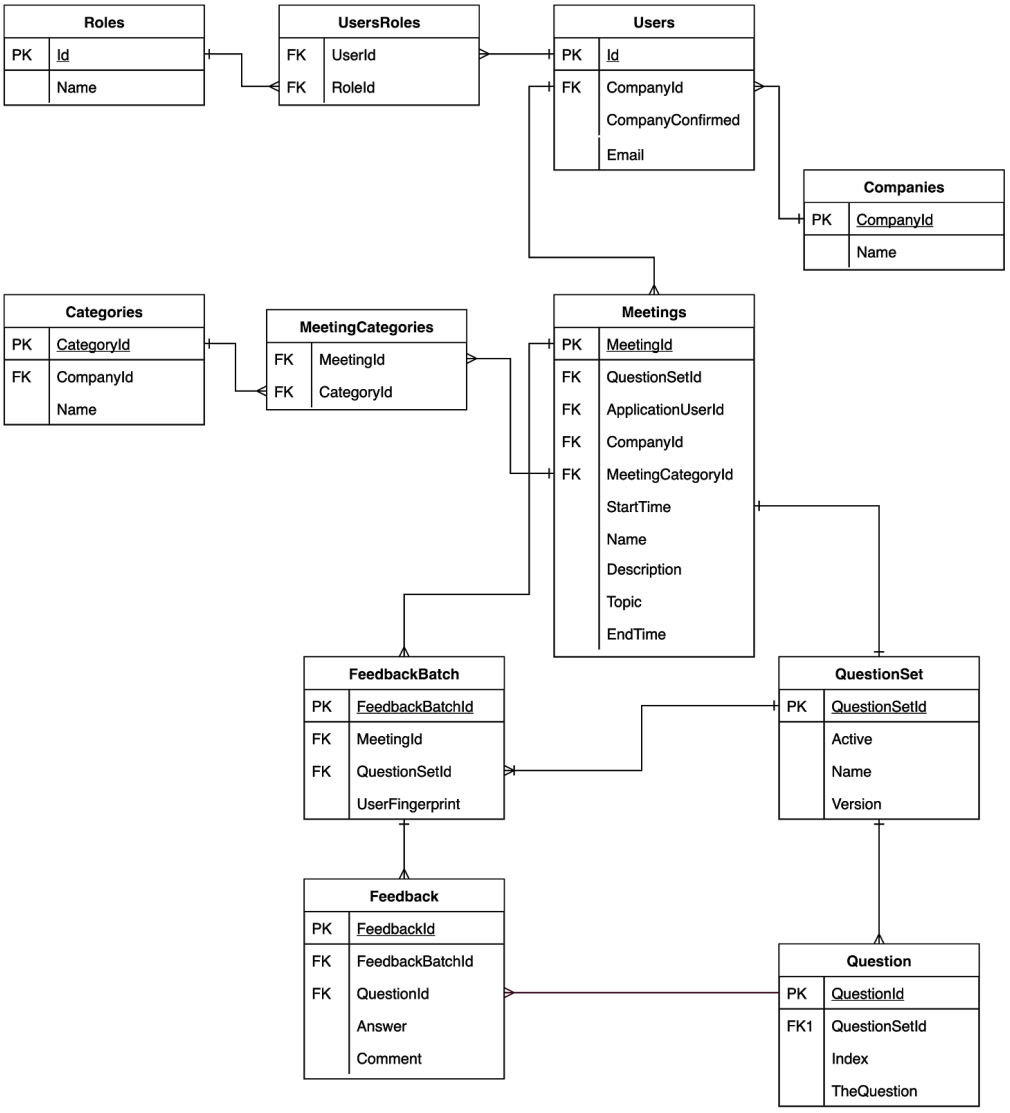
Som det også kan ses af figur 7, afhænger alle tre lag af WebApi, Business og Data af infrastruktur, som indeholder alle viewmodel/DTO, da disse bliver sendt frem og tilbage mellem alle tre lag.

Datamodel

Med udgangspunkt i domænemodellen blev en relationel datamodel designet. Datamodellen har gennemgået små ændringer undervejs, men har bevaret den samme overordnede struktur.

Figur 8 viser data og relationer til hinanden, vær opmærksom på at dette er en simplificeret udgave, dette er gjort for at gøre diagrammet uoverskueligt. Et skema diagram som angiver den fulde tabeller samt datatyper kan findes under Bilag 16 - Database skemadiagram.

Nogle af de relation som domænemodellen anviste modelleres nu som relationer mellem tabeller. Eksempelvis bliver mødeholderen nu modelleret som en “User” med en facilitator-rolle og tilknytter fx. stadigvæk til sine egne møder. Dvs. der findes en en-til-mange relation fra “user”-tabellen til “meetings”-tabellen.



Figur 8 - Entity Relationship Diagram

Persondataregulering

Databeskyttelsesforordningen[23] (herefter GDPR) udgør en væsentlig retlig regulering af, hvorvidt og i hvilket omfang persondata kan behandles. Det er nødvendigt at indtænke dette ved designet af kommercielle it-systemer. Der skal således tages højde for persondatabeskyttelse allerede i udviklingen af software, jf. GDPR art. 25 om Data protection by Design. Efter GDPR er der tale om persondata, når en oplysning kan henføres til en identificeret eller identificérbar fysisk person, jf. GDPR, art. 4, nr. 1. Behandlingsbegrebet fortolkes bredt, jf. art. 4, nr. 2.

Ved oprettelsen af en bruger i appen sker der en indsamling og registrering af persondata fra brugerne, og disse opbevares i en database med lokation i Nordeuropa. Der er tale om navne, telefonnumre og passwords. Disse udgør alle almindelige ikke-følsomme persondata efter GDPR art. 6. Passwords kræver dog fortrolighed, hvilket betyder, at disse oplysninger nødvendiggør en bedre beskyttelse, og dette stiller krav til anvendelsen af sikkerhedsforanstaltninger efter art. 32.

Som før nævnt bliver passwords ikke opbevaret i clear-text, men tværtimod som en hashet værdi, og dermed opbevarer systemet ikke brugerens egentlige password. I GDPR's forstand er dette udtryk for en psydonymisering af data, jf. art. 32, stk. 1, punkt a. Dette udgør en sikkerhedsforanstaltning, der skal imødegå integritetsrisici ved behandlingen af oplysninger om de registrerede.

Når oplysninger lagres på en server i Nordeuropa, er der tale om en overførsel i lovmæssig forstand, fordi oplysningerne derved bliver behandlet uden for Danmarks grænser. Det må dog forudsættes, at der er tale om en overførsel til et andet EU-land. Ved persondataoverførsler inden for EU må landene hverken indskrænke muligheden eller underlægge overførslen særskilte krav, jf. art. 1, stk. 3.

For at kunne opfylde kravene til oplysningspligt i henhold til GDPR art. 13 og 14, skal der også ved brugerregistrering fremgå en tydelig formulering af vilkår og data opsamlet fra en bruger, herunder behandlingsformål og retsgrundlag.

Der er forsøgt at tage højde for GDPR i udviklingen af systemet. GDPR er dog en kompliceret affære, og det kan derfor ikke udelukkes, at der findes implementerings detaljer, som ikke er efterlevet.

[23] ("Databeskyttelsesforordningen" n.d.)

Implementering

Både frontend og backend er udviklet i IDE'en Visual Studio Code. VS code blev valgt da der findes en lang række extensions til både C# og React som letter workflowet betydeligt. Derudover er det en åbenlys fordel ved ikke at skulle skifte mellem flere IDE'er.

Frontend

I dette afsnit vil beskrive centrale dele af implementeringen af PWA'en også kaldt frontenden i denne rapport.

Gennem dette afsnit vil både frontend og PWA blive brugt, hvor frontend vil blive brugt til at henvise til scenarier der ikke direkte har noget at gøre med om siden er en PWA, og omvendt vil PWA blive brugt i de sammenhænge, hvor det knytter sig direkte til denne teknologi.

Et boilerplate projekt[24] med basal opsætning blev brugt som udgangspunkt for udviklingen.

Da frontenden delvist er serverside genereret (SSR), findes der både dele af kodebasen, som kører på serveren og hos klienten, og dette vil så vidt muligt blive gjort klart gennem afsnittet.

Termer som useState og useCallback vil blive brugt til at beskrive den implementerede løsning i det kommende afsnit. Disse henviser til såkaldte "hooks", som findes i Funktionale React komponenter.

Store dele af frontenden er skrevet som funktionelle komponenter dette skyldes at funktionelle komponenter har en tendens til at være færre linjer kode og mere overskuelig at læse end klasse komponenterne, det er også anbefalet af udviklings team bag React, at anvende funktionelle komponenter så vidt muligt[25].

Der er ikke blevet valgt at bruge et enligt UI-komponent bibliotek, i stedet er der brugt særskilte komponenter. Grundlægende er der to årsager til at et komponent bibliotek ikke blev valgt fra start.

- Det var ikke ønsket fra kundens side at brugergrænsefladen skulle være for generisk af udseende (Ligne endnu en Bootstrap side).
- Udvalget af gratis komponent biblioteker som formår at UI mæssigt fungere fornuftigt på både desktop og mobil er meget begrænset. Der findes eksempler som Ionic[26], som har fokus på mobile Komponenter, men desværre fungerer Ionic's komponent

[24] (mvllow n.d.)
[25] ("Introducing Hooks – React" n.d.)
[26] (Ionic n.d.)

bibliotek på nuværende tidspunkt ikke sammen med Next JS. Desuden fungerer Ionic's komponenter også bedst hvis man bruger hele deres framework.

Dette har sløvet udviklingen af app'en væsentligt. Jeg havde en forestilling om at det var nemmere og finde mobil komponenter, end det endte op med at være.

Projekt opsætning

Det kommende afsnit vil beskrive opsætningen af udviklingsmiljøet brugt til at udvikle denne applikation.

Projektet anvender Typescript og transpiler til es6 JavaScript. Konfigurations muligheden *experimentalDecorators* er slået til (sat til true) for at kunne anvende annotationer i forbindelse med Mobx data store implementeringen, dette er beskrevet i detaljer i Data stores.

Type deklarationer til Cypress testing biblioteket er tilføjet under *types* for at Typescript kan give korrekt type-checking.

Hele typescript configuration kan findes i roden af projektet i filen *tsconfig.json*.

Som linter anvedes ESLint. ESLint anvender airbnb style guiden og kodeformatring er opsat med Prettier.

Web manifest opsætning

PWA'en anvender Web App Manifest specifikationen til selvbeskrivelse. Således indeholder denne PWA en manifest fil, der er lokaliseret i public mappen, da manifestet skal være tilgængeligt som en statisk ressource.

Manifestet giver alle de essentielle oplysninger om appen i JSON format og skal være tilstede for at være installerbar.

Det er også her navnet på appen er angivet og dermed det navn som vil blive brugt når appen er installeret og vil blive brugt sammen med ikonet. Ikonet i flere forskellige opløsninger er ligeledes lagt i public og refereret i manifest filen.

En PWA kan køre i forskellige display modes ud fra hvor meget af den bagvedliggende UI man ønsker at se.

PWA display modes[27]

Denne PWA anvender display mode *standalone* da det ønskes at basal funktionalitet såsom on screen hjem knap, tilbage knap osv. bliver vist. Der findes fire forskellige display modes at vælge fra fullscreen, standalone, minimal-ui og browser hvoraf den sidste vil køre appen som var den blot åbent i browseren.

Der angives også en url for hvor app'en skal starte når app'en bliver åbnet under keyen *start_url.*

Derudover angives et navigation scope, som siger noget om hvilke dele af siden, der kan blive opfattet som selve appen. I dette tilfælde er alle sider omfattet af appen, og der derfor sat til roden.

En tema farve angives også til app'en hvilket bliver brugt forskelligt alt efter hvilken platform og browser der bruges, eksempelvis bruges farven i top baren ved installation i chrome browseren.

Caching

PWA'en anvender caching for at muliggøre offline tilgang og forbedre performance markant. Til at konfigurere dette anvendes next-pwa[28], som benytte workbox[29] til at implementere og gennerere service workeren. PWA'en benytter precaching før installationen, hvor service worken cache'er alle kritiske statiske resursor[30]. Workbox står ligeledes for versions styringen af de cache filer og opdaterer disse ved ændring af en given ressource, dette gøres ved hjælp af runtime caching. Denne app anvender blot next-pwa default cache strategier configuration[31], med en lille ændring i den regex som angiver caching strategien for API URL'en.

Denne PWA anvender således tre cache strategier

- Cache first

[27] (Firtman 2020)
[28] ("Npm: Next-Pwa" n.d.)
[29] ("Get Started | Workbox | Google Developers" n.d.)
[30] ("Precache Files | Workbox" n.d.)
[31] ("Precache Files | Workbox" n.d.; shadowwalker n.d.)

- Stale while revalidate
- Network first

Nærmere beskrivelse af strategierne samt brugen af dem findes i Bilag 22 - Caching strategier.

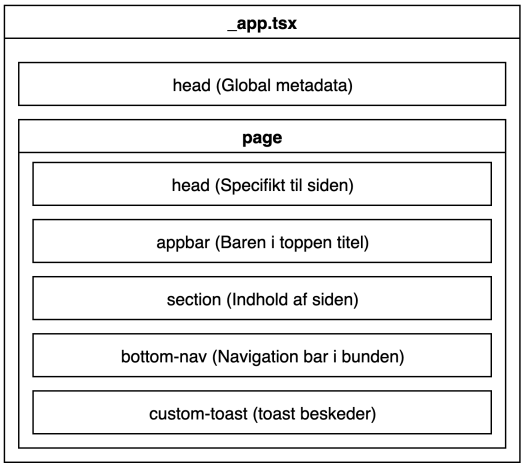
Cachings *indvirken på brugeroplevelsen*

Caching af PWA'en gør at brugeren oplever en mere responsive UI, som bedre kan matche en native applikation. Eksempelvis bruges flere steder i appen til animationer drevet af Lottie [32], som anvender json filer til at beskrive animationerne, som oftest er relativt store filer, man normalt vis ville være påpasselig med at bruge på en hjemmeside. Selv med en caching strategi på plads er filstørrelse på assets selvfølgelig stadigvæk en vigtig faktor, men dog i langt mindre grad.

Eksempelvis vil Lottie JSON filen blive gemt, og brugeren vil typisk ikke opleve nogen forsinkelse eller andre performance problemer, da dataen vil være til stede og med *Stale while revalidate* bliver opdateret, så snart klienten får adgang til en potentiel ny udgave på serveren.

Side anatomi

Sider følger i høj grad samme grund opbygning. Alle sider bliver rendered som børn af rod-komponent *_app.tsx*. *_app.tsx* sætte global metadata. Selve siderne består af metadata som se specifikt til kun denne side, derudover har en en *appbar*, som viser titel tilbage knap osv. *Appbaren* kan vises eller ej ud fra props fra page komponenten, dette gælder ligeledes *bottom-nav* som er menuen. Siden kan desuden indeholde sektioner hvis child-elementer vil være sidens indhold. *Custom-toast* er inkluderet her, da der er rigtig mange af siderne som benytter sig af toast beskeder, hvilket gør, at der ikke på hver enkelt side skal indsættes den komponent. Denne opbygning er visualiseret i figur 9.



Figur 9 - Side Anatomi (Ikke standart UML)

[32] ("Npm: React-Lottie" n.d.)

Styling

Alle stylesheets som skal anvendes globalt imported i *_app.tsx,* som NextJS forskriver[33]. Som før nævnt er det svært at designe og style komponenter, så de fungerer i et møde på mobil og desktop.

Til enheds-specifik styling benyttes *env()*[34], hvilket giver user-agent defineret variable. Således kan *safe-area-inset-top*[35]*,* som så kan anvedes til at placere eksempelvis en header.

På den måde kan udseendet af siden ændres så det passer til den specifikke enhed, og undgå at indhold eksempelvis er gemt bag UI fra operativsystemet.

global.css indeholder global styling, som ikke er defineret af externe stylesheets, dvs. stylesheets, der bliver importeret globalt, da deres styles bruges af komponenter, som ikke har self-contained styling.

Frontenden anvender *custom css properties*[36], dette giver mulighed for at have globale værdier til at justere designet. Dette er gjort for at skabe konsekvent styling af komponenter og gør det dermed også nemt at ændre dele af designet globalt, hvilket er, hvad der udnyttes ved temaskift fra mørkt til lyst tema. Langt de fleste *custom css properties* høre under :root[37]. root:, således at de kan tilgået i det globale scope for siden.

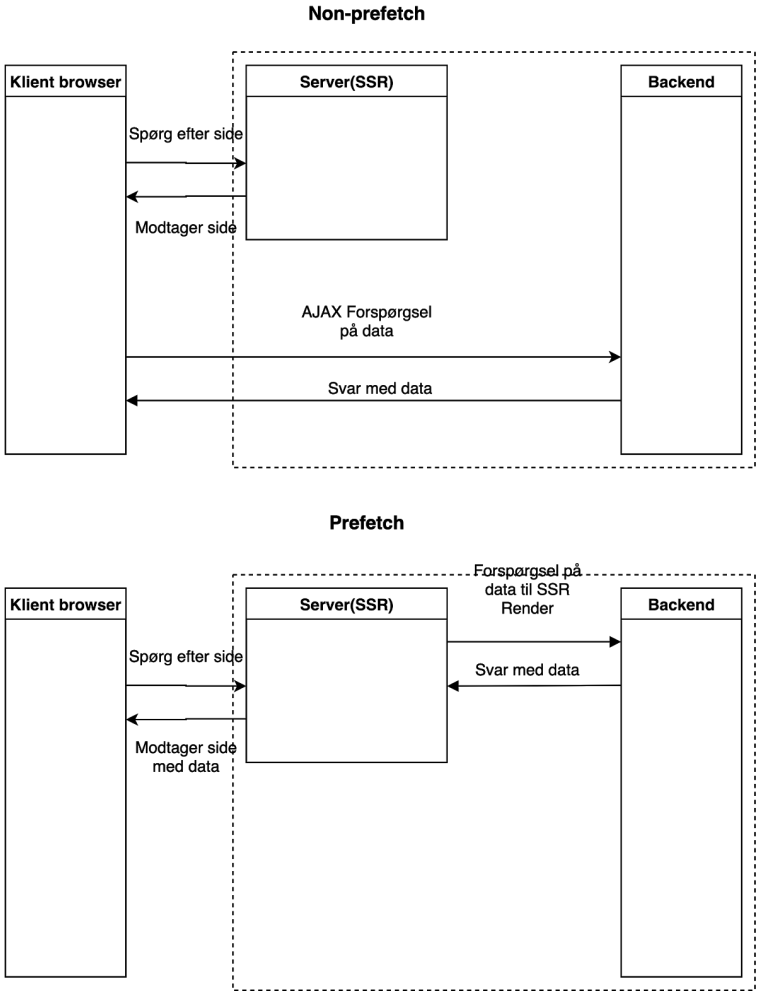
Styling på komponentniveau er så vidt muligt gjort, således at hver enkelt komponent indeholder den styling som er unik for det specifikke komponent.

Server side rendering med prefetch

Next js i modsætning til en CSR Applikation anvender SSR til prerender siden inden klienten modtager den. SSR gør derfor også at sider kan udfyldes med data fra backenden ved render på Next js serveren.

Det vil sige at der foregår server til server kommunikation hvor Next Js server agere klient til backend server. Denne kommunikation er vist i figur 10 men er udeladt i nogle af de kommende diagrammer, for ikke at øge kompleksiteten unødigt og fjerne fokus fra de centrale pointer.

[33] (“Learn | Next.js” n.d.)
[34] (“Env()” n.d.)
[35] (“Designing Websites for iPhone X” 2017)
[36] (“Using CSS Custom Properties (variables)” n.d.)
[37] (“Selectors Level 4” n.d.)



Figur 10 - Prefetch vs non-prefetch (Ikke standart UML)

Frontenden benytter både prefetch og klient-side ajax kald været brugt til at hente indholdet til sider.Mens SSR af komponenter er at foretrække, kan serverside fetching af data være et problem for at web application som en PWA hvor brugeren (Fordi der ingen browser UI er), forventer der sker noget, så snart der er blevet trykket på eksempelvis et link til en anden side. Ved at lade Next js stå for hente dataen er der altså intet der indikerer over for brugeren at siden er ved at loade hvilket kan være frustrerende. Derfor anvendes CSR i høj grad i tilfælde hvor payloaden er større, (TTFB bliver længere) og det derfor er mest hensigtsmæssigt at vise en form for loader i UI'en.

Data stores

Der findes en lang række stores i projektet, de er alle organiseret under en *rootStore.* Rootstorens opgave er at instantiere alle stores, som skal bruges, samt opsætte de stores, der bør være persistente og ikke mindst wrappe hele rootStoren i en React context.

Det er blevet gjort sådan, da dette gør, at man ude i hver komponent ikke behøver at bruge flere useContext[38] for at anvende flere stores. For at få adgang til flere forskellige stores og

[38] ("Hooks API Reference – React" n.d.)

deres variable kan objekt dekonstruktion[39] blot bruges til at udpege, hvilke store der er ønsket, for på den måde at have nem global tilgang til alt data i hele komponenttræet.

Observerer ændringer i data

Såfremt siden som anvender *rootStorens* kontext skal være reaktive, skal den attribut, som man ønsker at observere være decorate med annotationen @observable[40]. Derudover skal den givne komponent, som tilgår storen wrappes i mobx-react observer komponent.

State management og persistent lagring

Til at holde styr på applikationens globale stadie bruges biblioteket Mobx[41]. Til lokalt state i de enkelte komponentet anvendes useState hook'en[42], i og med langt de fleste komponenter er skrevet som funktionelle komponenter.

Der findes en lang række state management biblioteker, hvor Redux nok er det mest kendte. Jeg valgte Mobx, da det er simplere at sætte op og introducerer langt mindre boilerplate kode, end Redux eksempelvis gør.

I nogle tilfælde er det også påkrævet at persistere data, således at brugeren ikke oplever at miste et igangværende stadie, når applikationen bliver smidt ud af hukommelsen eller skal sætte sine indstillinger hver gang. Til dette anvendes mobx-persist som så står for at serialisere og persistere alt data som at annoteret @persist, dermed er det ikke nødvendigt eksplicit at gemme data i localStorage eller en anden form for persistent lager.

Data bliver blot gemt i localstorage som opsætningen er nu, men giver mulighed for at bruge LocalForage[43] for fremtiden hvis nødvendigt.

Persistent data hos klienten

Det er ikke nødvendig at skelne mellem stores som indeholder persistent data eller ej. *rootStoren* sørger for at hydrere de stores som indeholder persistent data.

Dette gøres ved at at kalde funktionen doHydration *rootStore* konstruktøren for på den måde at sikre, at alt data bliver data, som bør være i sync med en eventuel persistent udgave. doHydration tjekker også om den bliver invoked klient eller server side og kører selvfølgelig kun hydration i tilfælde af at den er hos klienten hvor localstorage er tilgængelig.

Det er nødvendigt kun at hydrere de stores, som skal indeholde persistent data og derved anvender @persist på mindst en attribut. Ellers vil mobx-persist smide en exception.

---

[39] ("Selectors Level 4" n.d., "Destructuring Assignment" n.d.)
[40] ("Modifiers · MobX" n.d.)
[41] ("Introduction · MobX" n.d.)
[42] ("Using the State Hook – React" n.d.)
[43] ("localForage" n.d.)

Hentning af data

Da stores'ne primært indeholder data som fetches fra REST API'en har selve storen også ansvar for at hente dataen i alle tilfælde, hvor der hentes data på klient siden.

Storene importer *ApiRoutes,* som navnet antyder indeholder alle router til ressourcer, som bliver brugt af frontenden.

Til alle Ajax kald bruges Fetch API'en[44]. Fetch Api'en har dog det problem at det ikke er support af legacy browser såsom Internet explorer. Dette kan dog nemt løses ved at anvende isomorphic-unfetch, som allerede er installeret i projektet, da den bliver brugt til server side fetching.

Hver store implementerer *IStoreFetchState* interfacet som indeholder *fetchState* atributten, der er af typen *FetchStates,* som er et enum, der bruges til at afgøre, i hvilke stadie et givent netværks kald er i.

Derudover indeholder *IStoreFetchState* også en attribut msg som til enhver tid indeholder beskeden dvs. status teksten fra det seneste ajax kald, dette bruges primært til debugging.

Modeller

Når JSON data'en bliver parsed til objekter bliver de eksplicit givet en type, således at de fremgår som denne type når storen gemmer data. Alle modeller matcher derfor med en DTO/POCO som findes på serversiden. Alle modeller findes i mappen *src/models.*

Rettigheder og roller

Rettigheder er som bekendt i denne application styret med roller. I frontenden er der derfor blevet implementeret en række tiltag for på nemmeste vis at undgå at brugere får adgang til uhensigtsmæssige sider, hvor der er indhold de ikke har ret til. Hvis brugeren alligevel skulle få adgang til sådanne sider vil backenden aldrig returnere indholdet til klienten alligevel, så gør dette blot at brugeren ikke vil blive præsenteret for en side som aldrig bliver udfyldt med data.

For at opnå denne styring har jeg valgt at wrappe alle sider, som kræver at være logget ind med et Higher-Order Component[45] (HOC) ved navn *withAuth*. Dette HOC står for at tjekke roller i *authStore* ud fra token, således at der nemt andre steder kan implementeres logik som anvender rollerne uden at skulle tilgå og parse token med rollerne hver gang. Der står den også for at redirecte til loginsiden, hvis token ikke er til stede (logget ud), eller token ikke længere er valid. Ved sider som forbeholdt en et subset af roller kan et array med disse roller angives som parameter til *withAuth.*

[44] ("Fetch API" n.d.)
[45] ("Higher-Order Components – React" n.d.)

Redirect til login siden bliver håndteret server side (Next js serveren) og derfor påbegyndes renderingen af siden slet ikke.

Hvis appen tilgås mens der findes en valid token på klienten vil et HOC, *withRedirect* omvendt føre en til hjemmeskærmen. Begge HOC lader, blot alle props er parser, alle props videre til selve siden.

Unikke tilbagemeldinger

Når en bruger afgiver en tilbagemelding på et event skal det kunne gøres anonymt, og man skal således heller ikke være logget ind da dette ville opstille en unødvendig barriere for brugeren som skal give tilbagemeldingen, da denne person måske ikke har en konto på tjenesten.

Samtidigt skal brugeren ikke kunne afgive mere end en tilbagemelding pr. event. Det derfor nødvendigt at kunne identificere brugeren unikt uden brugerne er logget ind.

Til dette har jeg valgt at bruge et "device fingerprint". Dette er valgt da det besværliggøre snyd og fingerprint-værdierne har lav risiko for kollisioner. Det løser dog ikke problemet at man blot kan anvende en anden enhed til at give feedback fra. Til at implamentere dette anvendes biblioteket *Fingerprintjs2*[46]*.*

Da der anvendes den gratis version af FingerprintJS er der mange oplysninger som undlades fra fingerprintet, dette betyder at selv om FingerprintJS er istand til at lave device-fingerprints, er det som reelt set anvendes her, med den gratis version, i højere grad et browser fingerprint[47].

Bemærkninger til enkelte sider

Alle sider befinder sig under src/pages. Dette afsnit vil beskrive implementeringen af ikke trivielle sider.

Routes er ikke eksplicit angivet noget sted men er i stedet et resultat af mappehierarkiet som findes under /pages. Det vil sige at index.tsx vil blive roden på siden. Sider som benytter sig af mere komplekse routing teknikker vil blive beskrevet, der hvor de bliver brugt.

Login

Da siden efter login processen er fuldendt, altid vil være /home, prefetches home siden for at formindske ventetiden og højne brugeroplevelsen.

[46] (fingerprintjs n.d.)
[47] ("Pro vs Free Version" n.d.)

En dag side

Lokaliseret under /pages/meeting/day. Siden modtager en dato i epoch format som en query parameter.

Next js routeren *(useRouter)* bruges til at udvinde selve datoen så den kan bruges til at hente møder for den pågældende dag via REST API'et.

For at implementere funktionaliteten som gør det muligt at skifte fra en dag til en anden på siden og stadig holde urlen up-to-date samtidigt med at undgå at måtte reloade siden for hver dato skrift anvendes shallow routing[48]

Den valgte dato bliver holdt i local state i komponenten og *useEffect* hook sørger for at forespørger de når datoen ændres og samtidig holde URL'en i sync.

Dashboard

Dashboardet er implementeret således at der er en række parametre som sendes til REST endpointed. Dette vil sige at i stedet for at få data, som er beregnet, med dette menes fx. modtager punkterne til grafen, modtager frontenden en version af data, som kun indeholder det som skal bruges til at generere de givne views.

Bemærkninger til komponenter

Grafer og diagrammer

For at producere de grafer som bruges til dashboardet anvendes react-chartjs-2[49], som reelt set bare er et wrapper komponent til brug af chart.js[50]. Valget faldt på chart.js da biblioteket har alle de typer af grafer som skulle bruges til dette projekt. Derudover er chart.js relativt simpelt og nemt kunne tilpasse til kundes behov.

QR-koder

QR koder bliver genereret i frontenden med npm pakken *react-qr-svg.* Dataen encoded i QR koden er den fulde url til feedback på et møde dvs. https://DOMÆNE/feedback/DVk. Dette er gjort for scanningen ikke blot kan bruges i selve appen, men kan scannes med hvilken som helst QR kode scanner, og stadig kunne sende brugeren til siden, hvor de kan give feedback.

QR-Kode læseren anvendes et allerede udviklet komponent[51], komponenten bruger WebRTC API'en[52] til at tilgå kameraet, og JsQR til at læse dataen fra QR koderne.

[48] ("Routing: Shallow Routing | Next.js" n.d.)
[49] ("Npm: React-Chartjs-2" n.d.)
[50] ("Chart.js | Open Source HTML5 Charts for Your Website" n.d.)
[51] ("Npm: React-Qr-Reader" n.d.)
[52] ("Getting Started with Media Devices | WebRTC" n.d.)

Da koden bruger WebRTC som ikke er til stede på server siden ved SSR bliver denne komponent nødt til at blive renderet klient side, dette gøres ved hjælp af en feature i Next js som muliggøre dynamisk import af en komponent, som det ville blive i en SPA[53] (Single page application)

Læg mærke til at for at IOS enheder kun kan bruge scanneren hvis enheden har IOS 13.4 eller senere[54]. Derfor skjules scanner knapper på enheder som ikke kan bruge scanneren.

Excel export

For at kunne tilbyde kunden en større frihed i forhold til at behandle feedback data, var det kundens ønske at lade dem eksportere den data som de har valgt gennem dashboard filteret.

Da der som beskrevet under dashboard allerede befinder sig de data som ønskes hos klienten, produceres excel filen blot i frontenden. For at opnå dette er der implementeret et komponent ved navn *excelExport.tsx* som visuelt blot fremstår med en download knap. komponenten anvender react-export-excel[55] npm pakken til producere selve filen og det kan angives i markup language hvordan filen skal udformes dvs fx. navnet på regnearket, antal og navne på kolonner osv.

Backend

Som før beskrevet er backend'ens primære opgave at udstille et API som frontenden kan bruge.

Identity framework bruges til at håndtere brugere i systemet. Dette betyder at alle operationer, som bliver eksekveret på en bruger foretages gennem Identity frameworks UserManager[56], RoleManager osv.

Den model som som bruges til at definere brugeren i database skemaet arver fra IdentityUser[57] classen.

Externe biblioteker

Det er anvendt en række forskellige eksterne biblioteker i backenden, disse er som følger:

- Newtonsoft[58] - Anvendes til serialisering og deserialisering af data til JSON data.

[53] ("Advanced Features: Dynamic Import | Next.js" n.d.)
[54] ("185448 – getUserMedia Not Working in Apps Added to Home Screen That Run in Standalone Mode" n.d.)
[55] ("Npm: React-Export-Excel" n.d.)
[56] (dotnet-bot n.d.)
[57] (dotnet-bot n.d.)
[58] ("Json.NET - Newtonsoft" n.d.)

- SendGrid[59] - Brugt til at sende mails over SendGrids Api.
- Automapper[60] - Mapping af objekter.
- Swashbuckle[61] - Swagger API beskrivelse og test.
- Entity Framework - ORM.
- Hashids[62] - Anvendes til encoding og decodeíng af møde id'er.
- SignalR[63] - Real-time kommunikation.
- ASP.NET Core MVC[64] - REST API og datavalidering m.m
- Identity framework[65] - Bruger og roller

Brugen af enkelte af disse biblioteker vil blive beskrevet mere i dybden igennem implementerings afsnittet.

Møde ID'er

Der har været et ønske fra kunden om at sikre en ikke åbenlys rækkefølge på møderne derfor bruger jeg er bibliotek kaldt Hashids[66] til at producere Unikke ID'er med et sæt udvalgte tegn.

Automatisk mapping af objekter

Ved brug af models og viewmodels vil der uundgåeligt være en del trivielle mappings mellem objekter. Et Auto Mapper bibliotek letter dette arbejde ved selv at kunne mappe mellem navne på klasse attributter. Dette kræver blot at man opretter en "profile" for hver de objekter, som beskriver hvordan der skal mappes fra input objekt til output objekt.

Langt de fleste mappings er mellem DAO objekter to DTO'er, dette vil typisk ske i et data-tilgangs-laget i et reposetoriesene således at alle objekter som bliver parset frem og tilbage gennem gennem de lag som bliver beskrevet under Løsningsarkitektur er DTO'er.

I tilfælde hvor der ikke er en til en direkte konvertering på key-value pairs mellem felter kan der således også anvendes mere komplekse mapninger. Fx. har dette været nyttigt ved mapping af møde. Ved møder foreskriver profilen automatik at mappe mellem heltal til Hashids representation, på denne måde bliver det gjort et sted i koden og er abstrahere væk fra alt andet logik, hvilket både gør koden nemmere at læse fordi, der ikke skal tages højde for den detalje og formindsker redundant kode.

Integration til mail afsendelse

Mails til fx. Aktivering af brugerkonto bliver gjort gennem Sendgrid's API, denne løsning er valgt da det gør, at der ikke skal opsættes en særskilt mail server, som skal vedligeholdes. Mail

[59] ("Email Delivery Service" n.d.)
[60] ("AutoMapper" n.d.)
[61] (zuckerthoben n.d.)
[62] (Brzóska n.d.)
[63] (bradygaster n.d.)
[64] (ardalis n.d.)
[65] (Rick-Anderson n.d.)
[66] (Brzóska n.d.)

funktionaliteten er wrappet i en service ved navn *EmailService.* Servicen anvender en Apinøgle til SendGrids service som opsættes som en miljøvariabel.

Sikkerhed

Dette system er der taget en række initiativer for at implementere basal sikkerhed i systemet.

En af disse tiltag er opbevaring af password som hashed værdier og salt'et værdi, frem for clear text, dette gør samtidigt at ingen ligger inde med det egentlige password og ved login af en bruger vil man i stedet for at sammenligne selve passworded med en værdi i databasen sammenholdes den hashed værdi persisteret i databasen samt den hash værdi for det potentiale korrekte login oplysninger og ved et match vil bruger blive tildelt adgang med en token.

En CORS politik er sat op således, at det kun er requests fra den specifikke frontend URL, som kan kan nå API'en, dette bliver beskrevet mere detaljeret under Cross-Origin Requests konfiguration i Bilag 21 - Udviklingsmanual.

Token baseret sikkerhed og login flow

For at sikre brugerne af systemets data har det været nødvendigt at implementere en form for sikkerhed. Valget er faldet på Token-baseret godkendelse da denne tilgang både er ganske fleksible og scalerbar i og med den er stateless, hvilket gør at det ikke er nødvendigt at opretholde en session store som ved brug Cookie-baseret godkendelse[67]. Nærmere betegnet anvendes en JSON Web Token, her fra forkortet JWT.

Et helt simpelt login flow er blevet implementeret. Flowet er således ud:

1. Bruger angiver login oplysninger som sendes i HTTP kroppen på et POST forespørgsel.
2. Serveren validere at brugerens oplysninger er korrekte og returnere en signeret JWT
3. Klienten gemmer token som en Cookie

Ved logud slettes cookien med JWT token blot på klientsiden.

Token er derefter nødvendig, at have inkluderet i header for at kunne får adgang til alle sikret endpoints. Requests som anvender SignaleR inkluderer dog ikke token i headeren, dette bliver beskrevet i Real-time tilbagemeldinger. Ved alle REST endpoints er angivet med annotationer, hvorvidt token validering er påkrævet, samt om en bestemt rolle skal være til stede i JWT'en, for at have adgang til det givne endpoint.

[67] (Kukic and Core n.d.)

Rolle baseret authorization

En af JWT'en andre fordele er, at man nemt kan gemme metadata også kaldet claims i den[68] . Som før nævnt anvendes Rolle baseret authorization, dette sker blot ved at en brugers roller er inkluderet i JWT'en, se Bilag 3 - JWT data (Læg mærke til at rollerne er repræsenteret i et JSON array ved mere end en rolle.). Således kan blive evalueret på både Klient og Server siden.
På klient siden bruges det blot til, at vise brugeren hvad de kan fortage sig og skjule funktioner, som de alligevel ikke har adgang til. Når det så er sagt ligger selve sikringen af data i backenden, for at sikre at det ikke er muligt at tilgå eller manipulere data, som man ikke har rettighed til at se eller ændre.

I skrivende stund er JWT'en signeret med en Symmetrisk signatur baseret på en secret da det ikke har været nødvendigt for platformen endnu at dele JWT med andre services[69].

Politik baseret authorization

Systemet indeholder en enkelt politik som validerer at brugeren har fået bekræftet sit tilhørsforhold til virksomheden, før personen får adgang til en lang række ressourcer.

Både rolle- og politik baseret authorization bliver blot en del af request pipelinen.

Data models validering

For at få Api'et til returnere brugbare fejlmeddelelser omkring evt. brug på datamodellen anvendes validation attributter[70] på DTO'erne, det betyder at når en regel, som en attribut foreskriver, ikke bliver anholdt vil api'en returnere med en fejl, som påpeger hvilke validerings exception som opstod.

Eksempelvis hvis et møde forsøges at oprettes uden navn vil en fejlmeddelelse retuneres da atributten "name" er anoteret med [Required][71], hvilket selvfølgelig betyder den ikke må være tom. Fejlen som Api vil returnere i dette tilfælde ligne dette:

```
{"errors":{"Name":["The Name field is required."]}, mere data..... }
```

På den måde undgås det at fejlen først sker ved mapping eller ved skrivning i databasen, hvor fejlen vil være svære at finde frem til.

Det har været nødvendigt at implementere en custom data attribut til at evaluere møde start og slut datoer op mod hianden.

[68] (Bradley, Sakimura, and Jones 2015)
[69] ("Ping Identity" n.d.)
[70] (dotnet-bot n.d.)
[71] (dotnet-bot n.d.)

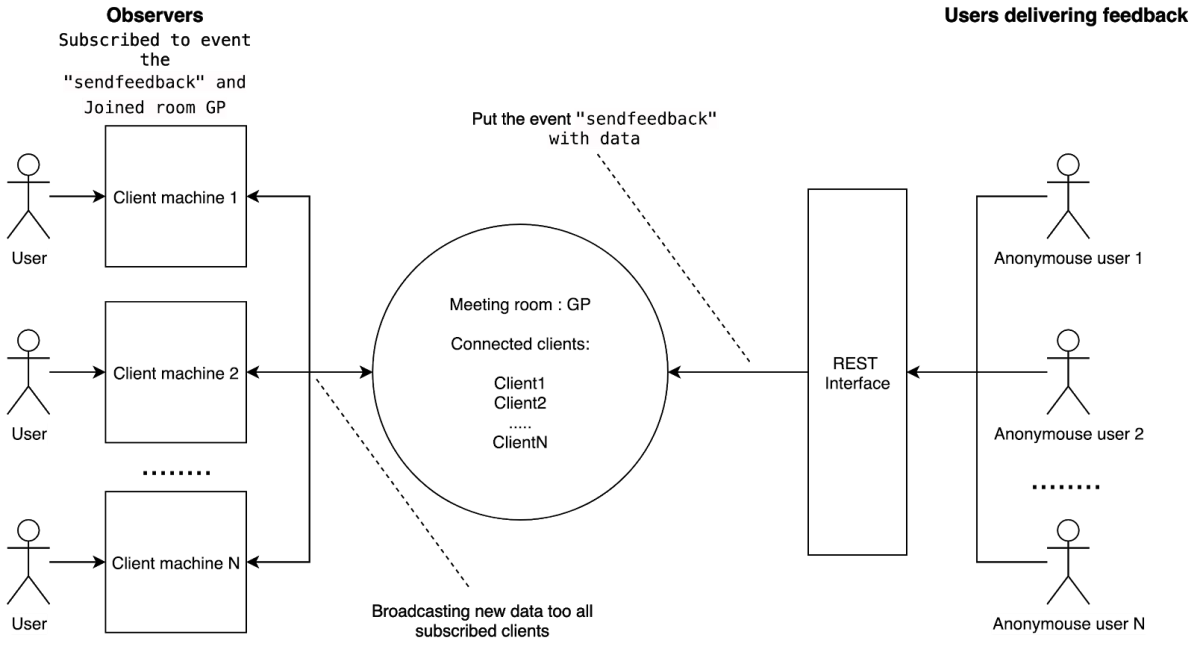
Real-time tilbagemeldinger

For at implementere real-time data er der på server siden anvendt et bibliotek kaldt SignalR, som gør det relativt nemt at arbejde med real-time data i .Net applikationer[72]. Derudover er der anvendt et klient bibliotek til SignalR[73] i frontenden.

SignalR faciliterer en nem implementering, samt tilføjer en række fallback teknologier således, at flere enheder kan anvende real-time data i app'en. SingleR anvender således disse teknologier i prioritetsrækkefølge:

1. WebSockets
2. Server-Sent Events
3. Long Polling

Systemet er implementeret således at der på serversiden findes et "rum" pr. møde som klienter kan subscribe til og dermed modtage tilbagemeldinger fra, dette er grafisk repræsenteret i figur 11. Dette opnås ved hjælp af grupper i SignalR[74], hvor gruppens Id er det samme, som det givne mødes short-id (Hashids).



Figur 11 - (Dette diagram er ikke standart UML)

Selve Join processen gøres i en RPC lignende facon hvor en metode kaldet *JoinRoom* som findes på serversiden. Protokollen er vist som et sekvens diagram i figur 12.

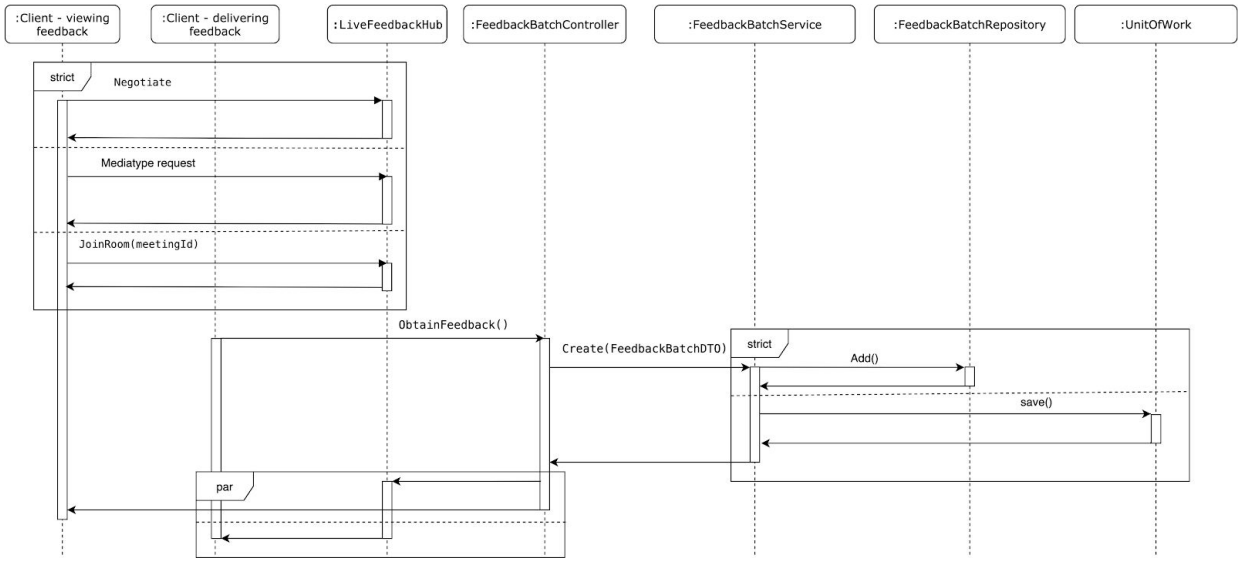
[72] (bradygaster n.d.)
[73] ("Npm: @aspnet/signalr" n.d.)
[74] (bradygaster n.d.)

Tilbagemeldingen gives gennem REST interfacet og ved hver tilbagemelding givet broadcastes de nye data ud til alle klienter i det givne møderum.

Selve endpoint'et er implementeret i *LiveFeedbackHub.cs* i WebApi projektet under Hubs mappen. Endpoint'et *"/liveFeedback"* anvender lige som REST endpoint'ene ligeledes JWT token til at identificere og regulere brugens adgang til data.

Alle brugere med en valid token kan oprette forbindelse til endpoint'et, men i det man tilslutter sig det specifikke møderum begrænses adgangen så det kun er brugeren, som har oprettet mødet samt virksomhedsadministration for den pågældende brugers firma, som har adgang til data'en.

Det har vist sig nødvendigt at sende JWT access token som en query string istedet for i headeren[75].



Figur 12 - Sekvensdiagram af real-time feedback protokollen

Dette er kun gældende for dette specifikke endpoint og det er dermed ikke muligt at parse token som query string med nogle andre endpoints.

For websocket kommunikation virker i produktion skal websockets slås til på Azure hvilket det ikke er pr. default. Azure har en kendt fejl, hvor man får statuskode 503 med websockets, ved handshake. Fejlen kan fixes ved at opgradere service plan og dernæst bare skifte tilbage igen[76].

SignaleR systemet understøtter alle de mest brugte browsere og internet explorer tilbage til udgave 11[77].

[75] (bradygaster n.d.)
[76] (MicrosoftDocs n.d.)
[77] (bradygaster n.d.)

Personlige score

Efter aftale med kunden er den personlige score udregnet på baggrund af de sidste ti møders tilbagemeldinger, og det er blot et gennemsnit af alle tilbagemeldinger givet på de ti møder, ganget med to. Der er ganget med to, for at skalaen som brugeren får præsenteret er fra 0-6 i stedet for 0-3, således at brugeren føler at der er et større spænd at udvikle sig over.

Controller- og Service-lag

Kontrollerne beskriver REST interfacet, som eksempel på hvordan kontrollerne anvender services tager figur 13 udgangspunkt i *MeetingControlleren*. Det derfor også kun *MeeringService* som er fuld udfyldt, for ikke tilføje unødig kompleksitet. Som det kan ses anvender *MeetingController* en række servies, som har til ansvar at implementere den business logik, som er nødvendig for at kunne udføre de request, som kommer gennem REST interfacet, eller hvilket som helst andet interface som ville kunne tilføjes.

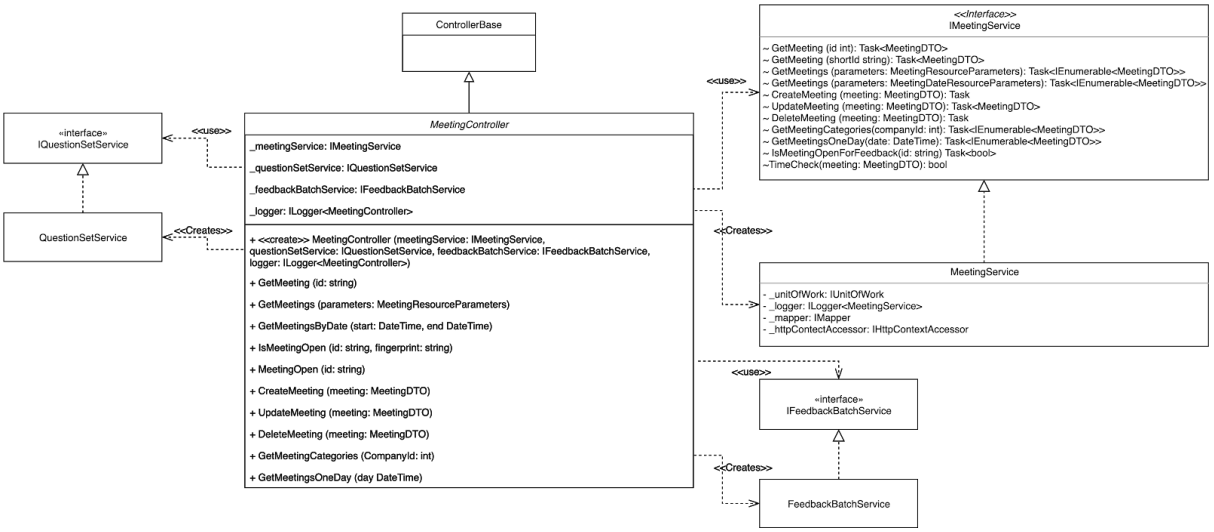


Figure 13 - controller brug af services

Service laget vil typisk anvende det unit of work til at udføre en operation på databasen, mappe/transformere data og for dernæst at returnere selv samme data.

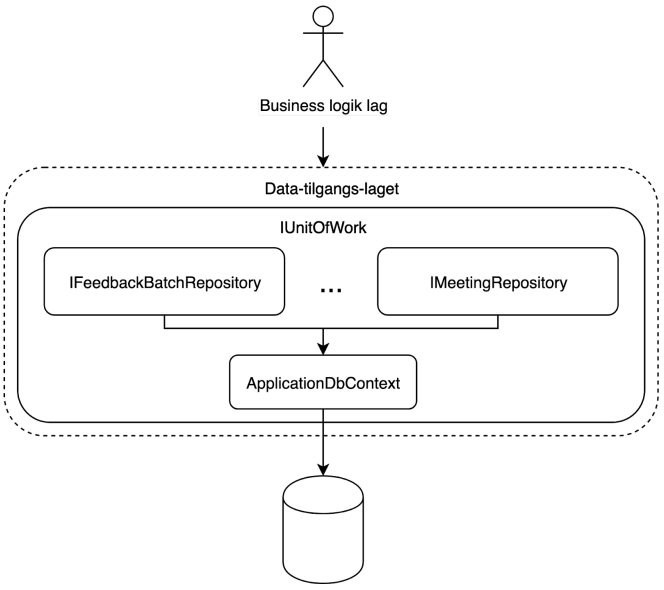
Data-tilgangs-laget

Data-tilgangs-laget er designet efter repository design mønstret[78]. Repository mønsteret afkobler det persistente datalager fra data-tilgangs-laget.

Derudover anvendes Unit of Work mønstret[79], som har til ansvar at centralisere adgangen til de forskellige repositories. Alle repositories tilgået gennem Unit of Work vil således også

[78] (nishanil n.d.)
[79] (Bibile 2019)

opererer under samme context og der kan foretages fulde transaktioner før der committees til ændring.



Figur 14 - Unit of Work i data-tilgangs-laget (Ikke standart UML)

Som overordnet filosofi, vil input parametre som bruges til at selected data blive presset helt ned til det repository, som håndterer det givne model. Dette er gjort for at kunne integrere parametrene direkte i querien, som bliver eksekveret på databasen. Dette betyder at man undgår at skulle håndtere unødvendigt store mængder data direkte i hukommelsen.

Database

Projektet anvender en object-relational mapper (ORM) til at interface med databasen og dermed anvendes den såkaldte code-first tilgang. Code-first tilgang gør det muligt at beskrive relational data på en objektorienteret måde og lægger dermed et ekstra abstraktionslag imellem data-tilgangs-laget og det egentlige data lag.

En ORM har den fordel at udvikleren i meget høj grad kan kan undgå at skrive rå SQL. I stedet manipuleres datastrukturer i hukommelsen, hvis ændringer overvåges af ORM'en og oversættes til SQL.

ORM'en har dog også visse ulemper. Den er god til at foretage simple CRUD operationer men ved mere komplekse operationer kan performance problemer opstå. Her vil det være nødvendigt at skrive SQL koden selv, dette kan dog nemt gøres, da det er muligt at skrive SQL direkte til SQL databasen som prepared statements.

I Entity Framework core defineres database skemaet ved hjælp af en kontekst som holder *DbSet* af alle de entiteter som skal findes i databasen. Relationer er defineret af kombination mellem konventionelle relationer som EF sammensætter på baggrund af attribut navne og annotationer direkte i entiteten, samt til mere komplekse relationer fluent Api'en[80]. Dette

[80] (AndriySvyryd n.d.)

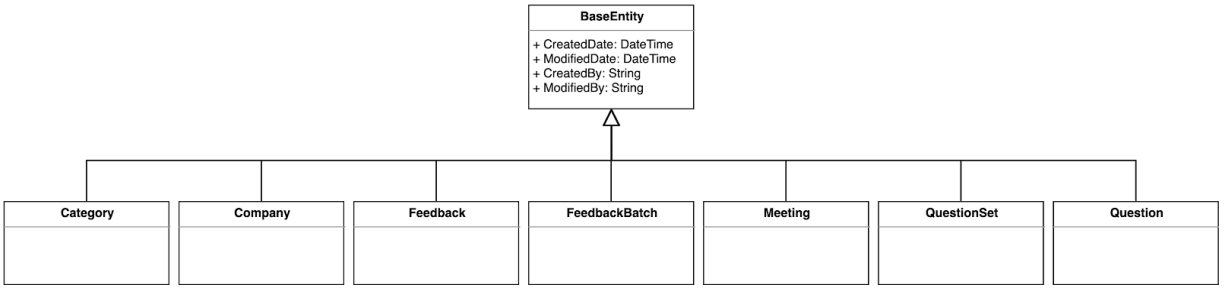
bliver omsat til DDL (Data Definition Language) af ORM'en og kan eksekveres på databasen.
Kontexten findes i *Data* projektet i mappen *Contexts* og filen *ApplicationDbContext.cs*.

Data traceability

I konteksten overskrives *SaveChangesAsync* samt *SaveChanges* dette gøres for at de kunne køre *AddTimestamps* metoden, som udfylder felter forbundet med den abstrakte klasse *BaseEntity.*

BaseEntity klassen indeholder følgende attributter

- CreatedDate
- ModifiedDate
- CreatedBy
- ModifiedBy



Figur 15 - BaseEntity arvehirakki

AddTimestamps udfylder disse felter. Dette gør at ændringer i dataen langt bedre kan følges.

Dette gøres ved at udnytte at EF core database kontext tracker entiteter, således kan det identificeres om en entity, skal modificeres eller oprettes i databasen, og dermed sætte et nyt timestamp på enden *CreatedDate* eller *ModifiedDate*.

Derudover kan koden i tilfælde, hvor der findes en HTTP kontext udlede bruger id'en ud fra de claims, som findes i den token, mere specifikt findes den i bruger id'en i claim med typen NameIdentifier[81]. I tilfælde hvor der ikke findes en HTTP kontext vil felterne *CreatedBy* og *ModifiedBy* blot markeret med "Anonymous".

Http konteksten vil være til stede ved alle kald til API'en som kræver Authorization dette kunne fx. være ved oprettelsen af et møde. Og den vil ikke være til stede ved afgivelsen af Feedback og derfor blot ved have "Anonymous"

[81] (dotnet-bot n.d.)

Bemærkninger til håndtering af data

Alle datoer er gemt som UTC datoer og det er dermed op til klienten at konvertere alle datoerne til den lokale tidszone. Dette virker til at være den bedste måde, at sikre at klienten selv kan styre hvilke tidszoner datoer ønskes i.

Når et spørgsmåls sæt "slettes" sker der ikke en enlig sletning i databasen, men bliver i stedet blot sat inaktiv. Dette sikrer at alle events, som har brugt dette spørgsmåls sæt stadig kan vise spørgsmålene, som der blev svaret på oprindeligt samtidigt med, at der ikke længere kan oprettes events med disse spørgsmål.

I tilfælde af at det underliggende database skema skal modificeres kan en migration foretages med EF Core, en mere detaljeret beskrivelse kan findes i Bilag 21 - Udviklingsmanual, sammen med en beskrivelse af data som bliver seedet i databasen ved første opstart.

Som supplement til denne dokumentation af API'en vil Swagger kunne bruges til at udforske den udviklet api. Swagger er blevet sat op så den kan modtage kan bruge JWT token.

Test og kvalitetssikring

For at sikre et robust produkt har systemet gennemgået en række test, her under Unit test, integrationstest og system test.

Produktet har ligeledes gennemgået brugertest for at sikre en intuitiv brugergrænseflade samt fokus på de rette features.Til fejlfinding er logging og monitoreringsværktøjer taget i brug.

En af de større ulemper ved at dette projekt er udviklet af én person er, at fejl kan blive introduceret en del nemmere ind i systemet. Normalt vil man lave et pull request til fx en staging branch, hvor koden ville gennemgå et code review af en anden udvikler, som vil kunne gennemgå koden og forhåbentlig finde alle de mest kritiske fejl. Det er selvfølgelig stadig muligt at lave code review selvom man er én udvikler men sandsynligheden for at finde de kritiske fejl formindskes væsentligt da det er ens egen logik, som man reelt evaluerer og derfor vil være tilbøjelig til endnu en gang anse det skrevne kode for at være korrekt.

Andre værktøjer brugt til test under udvikling

NgRok

For at kunne gøre det muligt at teste PWA'en på et tidligt stadie i udviklingen, dvs. før den enlige deployment har fundet sted er værktøjet NgRok[82] blevet brugt til at udstille min localhost til en offentlig tilgængelig URL således at den kan ses på mobilen.

Swagger

Swagger er blevet brugt under udvikling til at teste diverse endpoints i API under udviklingen for at sikre, at de hver især opfører sig som forventet og sender de rette data samt HTTP statuskoder tilbage til klienten. Swagger har ligeledes været et uundværlig væktøj til at teste authentication skemaet på på APIen således at de endpoints som skal være beskyttet er det.

I Swagger UI'en betyder dette også, at der skal indsættes en valid JWT token i Swagger for at kunne bruge langt de fleste endpoints. En valid token kan anskaffes ved blot at anvende authentication end pointet med en bruger oprettet i systemet.

Postman

I tilfælde hvor Swagger ikke har givet nok muligheder for at rette i selve requested eller tilfælde hvor et specifikt request fra frontenden har skulle replikeres en til en er Postman blevet anvendt for at debugge den uønsket opførsel.

Logging og fejlfinding

Logging bruges både i frontend og i backend til at finde fejl og monitore brugeraktivitet.

Logging i frontend med google analytics

I frontenden anvendes Google analytics til at opsamle data. Der er bliver lavet en række hjælpe metoder til at sende data til googles platform, disse findes i */utils/analytics.ts.* Første gang en bruger loader en side dvs. *page* komponenten vil React-ga som anvendes til at interagere med Google analytics. React-ga bliver initialiseretet med med en unik tracking code og et flag vil blive sat på en globalt scope declared variable på window, så init kun en gang.

Herefter kan *logPageView* metoden køres som blot vil window location og logge et besøg til den pågældende side. Dette gør at man nemt kan få et overblik på googles platform over hvilke sider folk besøger.

[82] (inconshreveable n.d.)

Ud over side besøg bruges google analytics til at logger events og fejl. Et event som eksempelvis bliver logget er hvordan brugerne åbner appen dvs. om appen er åbnet i en browser tab eller i standalone mode som appen vil køre, hvis den er installeret, således kan det ses ud af tallene, hvor stor en andel af brugerne som installerer PWA og anvender den som en enlig app.

Når en fejl opstår bliver den ligeledes logged til google analytics dette give udvikleren en mulighed for at opfange og rette fejl som skulle være sluppet igennem til produktionsmiljøet.

For dette projekt har det også været meget interessant at se hvilke enheder PWA'en bliver brugt på da dette kan give en indikation af hvilke platforme, som skal være target platformen for at give den bedst mulige oplevelse til flest mulige brugere, dette kan ses i bilag 6[83] sammen med en række andre nøgletal.

Logging i backenden

Backenden har opsat en logging provider, som udskriver i logfiler som kan findes i Azure. Derudover kan servicen nøje monitoreres gennem Azure's systemer, det vil eksempelvis sige at det kan ses hvor mange 404 responses servicen har returneret, samt mange flere parametre.

Uptime monitoring

Som led i at opretholde en stabil service over for test brugerne monitoreres hver 5 min hvorvidt både backenden samt frontenden er kørende. Frontenden tester den bare på roden af domænet og backenden er der lavet er health endpoint, som meget simpelt bare svarer tilbage med en tekst og 200 såfremt den kører. I tilfælde af en af de to ikke kører modtager jeg en mail så problemet kan løses hurtigst muligt. Til Dette anvendes Uptime robot[84].
I Bilag 12 - Uptime monitoreringsdata kan der ses mere.

Unit testing

Dette afsnit vil beskrive hvordan unit test er anvendt til at teste isolerede komponenter gennem de lag, som projektet består af.

Backend

Backend har en længere række unit test som har til opgave at test Service samt repositories og sikre at data bliver tilgået og manipuleret korrekt af de forskellige service som REST controllerne anvender. Selve REST interfacet bliver testet via end-to-end testing fra frontenden.

[83] Bilag 6 - Fordeling mellem mobile og desktop brugere
[84] ("Uptime Robot | Free Website Monitoring" n.d.)

Test projektet er sat således op at Entity framework i steden for at anvende MSSQL driver, opretter en in-memory[85] database som anvendes til alle test.

Dette er gjort for ikke at være afhængig af en forbindelse til en enlig kørende SQL server dette medfører selvfølgelig at selve datalaget, men derimod data-tilgangs-laget, der testes i disse unit test, ikke er den samme underliggende database implementation som anvendes i produktion.

Det giver også den fordel, at der ikke skal laves nogle clean up af databasen da den jo kun eksisterer mens testen er i gang.

Frontend

I frontenden anvendes Jest som Unit test framework testene har til opgave at teste services og centrale dele af kodebasen. Test coverage kunne bestemt have været bedre.

Integrations testing

Frameworket Cypress[86] anvendes til at foretage disse test. Hver test modsvarer en use case.

Læg mærke til at det er vigtigt, at testene bliver kørt på et production build da HMR[87] ellers vil kunne resultere i upålidelige test resultater.

Det er ikke valgt at inkludere integrations testene i deployment pipelinen, da dette ville kræve en en betalt konto hos cypress.

Fokuset for Integrations testene har været at opspore fejl i opførelsen af applikationen ved sammensætningen af delkomponenter. Der er således taget udgangspunkt i brugsscenarierne med særlig fokus på kerneydelsen i systemet.

Testscenarier

Testscenarie ID	Navn på test	Status
TC01	Brugerregistrering	OK
TC01-01	Brugerregistrering - Med eksisterende virksomhed	OK
TC01-02	Brugerregistrering - Med ny virksomhed	OK
TC02	Login	OK
TC02-01	Login med Admin bruger	OK

[85] (ajcvickers n.d.)
[86] (“JavaScript End to End Testing Framework” n.d.)
[87] Hot Module Replacement

TC02-02	Login med Virksomhedsadministrator bruger	OK
TC02-03	Login med Facilitator bruger	OK
TC03	Oprettelse af møde	OK
TC03-01	Oprettelse med minimum af information	OK
TC03-02	Oprettelse med flere kategorier	OK
TC03-03	Oprettelse med flere kategorier og beskrivelse	OK
TC03-04	Oprettelse uden navn (Negativ test)	OK
TC03-05	Oprettelse uden dato ændring (Negativ test)	OK
TC03-06	Oprettelse uden dato spørgsmåls sæt (Negativ test)	OK
TC04	Feedback afgivelse	OK
TC04-01	Load møde, aflæs ID og brug input felt med ID til feedback	OK
TC04-02	Mødet burde afvise feedback anden gang	OK
TC04-03	Mødet findes ikke	OK
TC04-04	Mødet er uden for den åbne periode	OK
TC05	Møde sletning og load validering	OK
TC05-01	Validere alle møde informationer bliver hentet	OK
TC05-02	Slet møde	OK
TC06	Spørgsmål sæt	OK
TC06-01	Tilføj spørgsmåls sæt	OK
TC06-02	Ændre i spørgsmåls sæt	OK
TC06-03	Slet spørgsmålssæt (Fra index)	OK
TC06-04	Slet spørgsmålssæt (Egen side)	OK

Der kan desuden findes en traceability matrix mellem brugsscenarier og testscenarier i Bilag 7 - Testscenarier og brugsscenarier matrix.

System testing er blot blevet foretaget manuelt, ved at gennemgå funktionaliteten virker tilfredsstillende på produktions serveren.

Brugertest

Gennem udviklingen af systemet er der blevet foretaget to brugertest.

Første brugertest

Denne test blev lavet med to Spinoff ansatte over videoopkald, det var desværre ikke muligt at gennemføre test sammen med brugerne på grund af corona pandemien.

Brugerne fik lov til at prøve appen og kommentere på de forskellige aspekter af applikationen.

Brugerne have ikke fundet frem til hvordan appen skulle installeres da de ikke før havde prøvet at installere en PWA og at den popup, som beder brugeren om at installere appen ikke var blevet vist.

Der var mange ændringer til layout og sproglige fejl. En detaljeret beskrivelse kan findes under Bilag 17 - Første brugertest.

Anden brugertest

Denne test var påtænkt at skulle foregå i samarbejde med to af Spinoffs kunder. Der var dog ikke blevet forventningsafstemt tilstrækkeligt, da kunderne troede de ville få lov at prøve et tilnærmelsesvis færdigt produkt.
Dette var ikke tilfældet og derfor meldte kunderne sig fra testen. For stadig at få bruger feedback blev flere Spinoff ansatte samt familie inddraget som testere. På grund af covid-19 blev testen desværre ikke foretaget således at udvikleren kunne observere dem under brug. Derfor blev test brugernes oplevelser og forslag skrevet ned som det kan ses i Bilag 9 - Anden bruger test.

Overordnet kode kvalitet

Da jeg i vid udstrækning ikke før har prøvet kræfter med mange af teknologierne brugt i projektet, har der undervejs været flere redesigns af kodebasen. Dette bærer kodebasen også præg af og der er brug for en refaktorering af dele af koden.

Koden blev meget kommenteret i mindre grad. Der er også flere steder i kodebasen (især) i backenden hvor ikke alle konventioner omkring arkitekturen bliver overholdt.

Der er blevet kun foretaget Unit test på dele af kernefunktionalitet og det er derfor min vurdering at der bør bruges noget mere energi på er verificere de forskellige komponenter virker tilfredsstillende.

Kendte fejl

Systemet har en række fejl som er blevet identificeret men ikke er blevet nået udbedret inden aflevering.

- Det kan ske ved beregningen af feedback at mobx laver en state update flere gange lige efter hinanden hvilket får resultatet til at blive 0 på tværs af alle spørgsmål.
    - Problemet ville muligvis kunne løses ved at lade udregningen foregå som en @computed attribut i MobX
- Hvis man scroller datovælgeren til sine grænser kan den producere datorer som overlapper til den næste dag.
- Kalenderen bliver større end den enlige skærmstørrelse, hvilket gør brugeren skal scrolle for at se hele kalenderen hvilket er et irritationsmoment.
- På nuværende tidspunkt kan det at tage et spørgsmål sæt i brug give uhensigtsmæssige bivirkninger. Dette kunne eksempelvis være at rækkefølgen på spørgsmål ændres således, at folks svar bliver parret op med et andet spørgsmål end hvad de har svaret på.
    - Et løsningsforslag kunne være, at undgå at ændre selve spørgsmål sættet ved en ændring, men i stedet for laver en kopi med et højere versionsnummer, hvor den gamle udgave samtidig bliver sat inaktiv.
      På den måde vil det hele tiden kun være den nyeste udgave som kan vælges når det oprettes møder men alt feedback givet på gamle udgaver vil blive vist rigtigt og data vil være konsistent.
- På nogle IOS enheder er det desuden blevet observeret at UI'en fryser til tider. Dette virker til at være fuldstændigt uafhængigt af hvor i app'en man befinder sig og grunden til dette er ikke fundet.

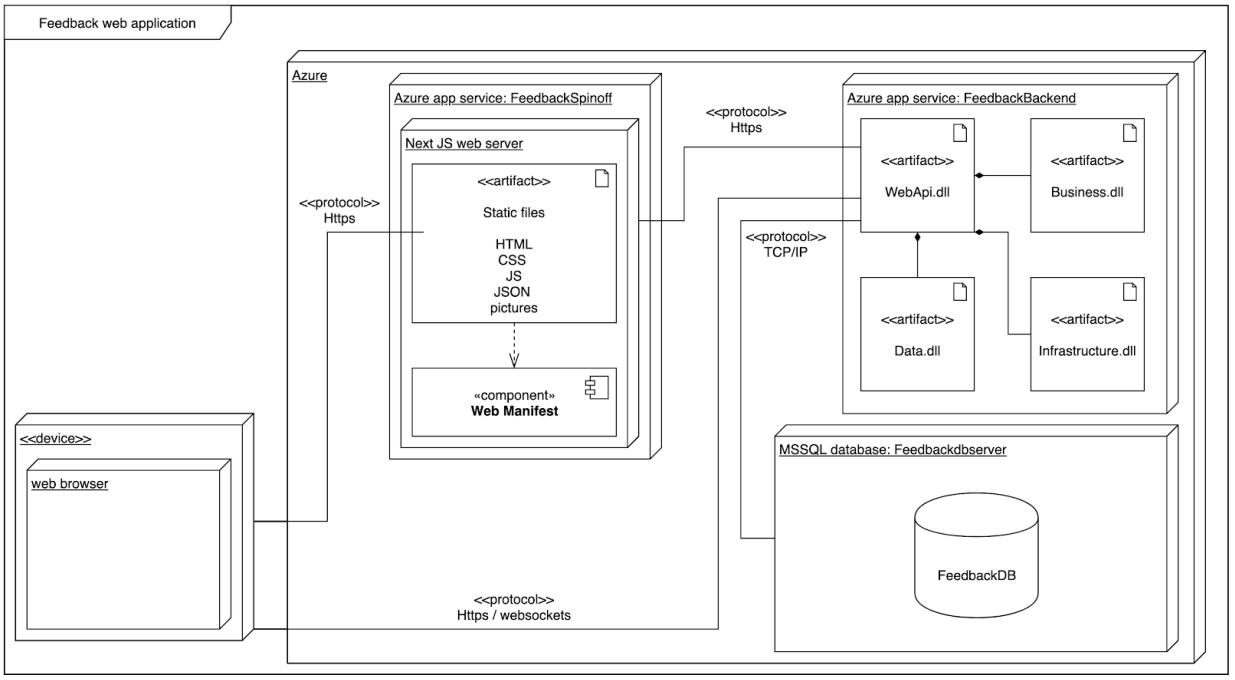
Deployment og drift

Dette afsnit vil beskrive opsætning af løsning på Azure. Derudover beskriver det hvordan contenues integration er sat op for projektet. En gennemgang af de specifikke miljøvariable som skal sætte for løsningen kan køre kan findes i Bilag 21 - Udviklingsmanual.

Deployment diagram

Instans niveau deployment diagrammet viser hvordan de enkelte software komponenter kører dvs. enhed/maskine samt container osv.

Nærmere viser dette diagram, hvordan systemet er opsat på Azure i skrivende stund.



Figur 16 - Deployment diagram - Opsætning på Azure

Både Frontend- og Backend-applikationerne er deployeret som en Linux-baseret *Azure App Services* som bruges til HTTP(s) baseret applikationer[88]. Det er vigtigt at alt Http kommunikation anvender SSL, dette er både fordi det er et krav for at kunne distribuere en PWA samt for at sikre at sensitive informationer såsom kodeord ikke findes som clear tekst i http payloaden.

Selve databasen er en MSSQL database, som serves af MSSQL Server i sin egen app service/container.

I og med at det er WebApi.dll som indeholder start konfigurationen er det også den som står for at oprette forbindelse med databasen.

De artefakter som er angivet under FeedbackBackend er dem som er defineret af løsnings forskellige projekter, der er en lang række yderligere artefakter som hovedsageligt er DLL filer for alle de biblioteker som anvendes af de samlet projekter, således findes eksempelvis et artefakt *AutoMapper.dll* som er associeret med Infrastructure.

Deployment pipeline

Projektet er fordelt på to repositories[89] begge er opsat således at projektet automatik deployes gennem en pipeline, som populært sagt hedder continuous integration, og ligeledes er der blevet deployet løbende så kunden kunne følge med i ændringer og nye features også kaldt continuous delivery. CI/CD er i høj grad også en filosofi og et workflow, hvilket kun i nogen grad er blevet brugt under projektet.

[88] (cephalin n.d.)

[89] Bilag 4 - Link til repositories

Jeg har brugt githubs egen pipline system ved navn Github actions, da dette kræver minimal opsætning for at køre tilfredsstillende. Selve pipeline definitionerne (.yml fil) ligger i roden på begge repositories under .github/workflows.

Dette kan tage noget tid at sætte op, så det fungerer korrekt men sparer tid på den lange bane da mange af de manuelle steps bliver automatiseret for på den måde hurtigere og nemmere at kunne teste og levere software løsningen.

Således compiler og bygger applikationerne på en virtuel maskine hosted hos GitHub, som github i deres dokumentation kalder en runner[90]. Dette både offroader processen fra den lokale maskine, der bliver udviklet på men også giver et nemt overblik over om projektet bliver kompileret korrekt.

Begge pipelines til både frontenden og backenden er sat op til at deploye løsningen ved push til master branchen. Derudover anvendes *ubuntu-latest* image'et til begge runnere.

Backend pipeline

Først i pipelinen bliver opsat .NET Core SDK'en med version 3.1.102 således at CLI'en den er til rådighed.

Derefter køres alle unit test cases for at validere at den nye version pushet bør fortsætte med at blive lagt op på Azure serverne.

Dernæst kan build køres, som kompilerer alle projektets egne filer sammen med alle dependent biblioteker, dvs at alle klasse biblioteker, som er beskrevet i Løsningsarkitektur afhængigheder også inkuderes. således produceres en række DLL filer som indeholder MSIL samt metadata[91].

For at kunne overføre og deploye koden skal projektet dernæst publish vil generere alle filer som er nødvendigt for at kunne køre applikationen på en anden maskine.

Alle disse filer overføres til App service containeren på Azure for at kunne lov til at gøre dette anvendes en såkaldt *publish-profile,* som er gemt som en secret på Github.

På selve den virtuelle maskine hvor App servicen kører findes at kompileret kode under */home/site/wwwroot* som DLL filer.

[90] (cephalin n.d.; "Virtual Environments for GitHub-Hosted Runners - GitHub Help" n.d.)
[91] https://support.microsoft.com/en-us/help/815065/what-is-a-dll

Frontend/PWA pipeline

Først opsættes Node js version 10, dernæst udskiftes next konfigurationen, uddybende forklaring findes i Miljøvariable og konfiguration dernæst køres npm install, build og test i et job.

Npm install, henter og installere alle npm pakker, build kompilere alle filer, som skal bruges for at kunne serve produktions version af applikationen.

Disse filer overføres dernæst til App service containeren på Azure der ligesom backend pipelinen anvender en *publish-profile,* som er gemt som en secret på Github.

Videreudvikling

Virksomheden har undervejs haft mange ønsker som ikke har været taget med i denne prototype jeg vil derfor her nævne nogle af deres ønsker og hvad jeg ser som de vigtigste næste trin at tage.

Login systemet er på nuværende tidspunkt meget simpelt og returner essentielt bare en JWT ved login og brugeren bliver sendt til login skærmen så snart denne token er udløbet. Det ville derfor være oplagt at implementere en refresh token således, at brugeren kan forespørge en ny access token.

Kunden har en lang række features, som de ønsker for fremtiden og det er min anbefaling at holde igen med mange af disse features til at produktet har været i en beta test med slutbrugere, som vil kunne vurdere kerneprodukten, før der implementeres yderligere features som måske kan fjerne fokus fra produktet kerneydelsen.

Når det så er sagt, er der en lang række ting, som absolut vil være værd at inkludere i produktet når alle kendte fejl er rettet. Jeg ville videreudvikle på systemet med denne prioritering:

Høj prioritet - Klargør til beta test med større brugergruppe

- Tilføje muligheden for at slette og oprette kategorier.
- Tilføje versionering af spørgsmåls sæt
- Refresh token
- U15 bør blive implementeret.

Mellem prioritet - Gør klar til at være et betalt produkt

- Gøre det muligt at arbejde på et spørgsmåls sæt og først offentliggøre det når det er færdigt.
- Betalingssystem med abonnement løsning.

- Outlook integration, som vil gøre det muligt blot at oprette evnet i en særskilt kalender i outlook og de vil automatisk blive oprettet i Opino.
  Microsoft har en række Api'er[92] som ville kunne bistå i udviklingen af denne feature.
- Google calendar integration
- Give grafer farver på baggrund af deres værdi (En lav score kunne være rød og en høj score kunne være grøn).
- Lade virksomhedsadministrator vælge specifik medarbejder i dashboard.

Lav prioritet - Features som kunne være nyttige for visse bruger segmenter

- Mulighed for at skrive en indledende besked før feedback afgives.
- Jeg vil mene det kunne være en god ide at investere nogle højkvalitets komponenter til Kalenderen og datovælgeren, da jeg vil tro disse begge ikke vil leve helt op til brugerens ønsker. Dette kunne eksempelvis være Mobiscroll's UI[93] bibliotek som i høj grad fokusere på den mobile brugeroplevelse.
- Mulighed for at vælge sprog.
- Notification til brugeren når mødet er åben for feedback.
- Mulighed for at Admin kan filtrere events på Virksomhed og bruger.
- Afdelingsrapport fordelt per. medarbejder.
- Gøre det nemt at kopiere og dele møde ID/QR kode.

Konklusion

Dette afsnit vil indeholde to delkonklusioner, en procesorienteret og produktorienteret konklusion.

Derudover vil der blive beskrevet hvilke muligheder for videreudvikling, som vil være en nødvendighed for systemets kommercielle validitet, eller interessante at forfølge for at forbedre produktet.

Procesorienteret

En del elementer i projektet blev ikke afdækket initialt, hvilket betød at projektet blev underestimeret. Projektet er dog kommet tilfredsstillende i mål, ved at evaluere fremgangen undervejs og foretage små justeringer af scopet.

Dokumentationen for scope ændringer og planer kunne forbedres. Der har været få mennesker involveret i projektet og det har derfor været nedprioriteret at nedskrive referater af hver enkelt møde.

Det har gennem projektet været forsøgt at få gennemført brugertest med "rigtige" slut brugere, hvilket kun til dels er lykkedes. Dette skyldes i nogen grad Covid-19 pandemien, da brugertesten måtte foretages remote, hvilket gjorde at meget værdifuld data gik tabt, i og

[92] (Archiveddocs n.d.)
[93] ("Cross Platform React Mobile UI Controls | Mobiscroll" n.d.)

med brugerne ikke kunne observeres mens de brugte systemet. Desuden manglede forventningsafstemning mellem Spinoff og deres testpersoner, hvilket gjorde de bakkede ud af testen.

Spinoff har dog igennem hele processen været gode til at give feedback og dette har gjort op for noget af det tabte, selv om det var ærgerligt at en “rigtig” brugertest ikke blev gennemført.

Gennem projekter har Spinoff haft en voksende liste af features de har ønsket. Der har måtte fokuseres på kerneydelsen for der ikke skulle opstå scope creep, dette er til dels lykkes , nogle features er dog blevet taget med på grund af den relative lille indsats, der skulle bruges på implementeringen.

Der har gennem hele projektet været en god kontakt med Spinoff, møder er blevet taget efter behov og jeg synes det har fungeret meget fint.

Projektet blev relativt tidligt i udviklingsprocessen idriftsat hvilket var en kæmpe fordel under i kommunikationen med Spinoff og ved system testing, da de aktivt kunne følge med og teste produktet.

Da jeg ikke har arbejdet med mange af teknologierne tidligere, har der været meget der skulle læres undervejs hvilket i den grad også har bremset projektet, men har også været en meget lærerig process.

Produktorienteret

Projektet har resulteret i en prototype, som ligger inden for den aftalte MVP med Spinoff.

Mangel af et endeligt UI Komponent bibliotek har sløvet udviklingen og har gjort det svært at lave rapid prototyping. Det vil være min anbefaling at se på et betalt UI bibliotek som fokusere på den mobile brugeroplevelse, da der er en række komponenter som ikke lever op til den de interaktionsmuligheder som findes på den Native platform. Under videreudvikling foreslås et konkret bibliotek.

Efter nu at have prøvet Next js til et PWA formål er mit overordnede indtryk, at det giver hurtige PWA’er, men at dette projekt havde været bedre stillet på den korte bane ved at vælge et framework som Ionic, som netop har fokus på at emulere den native mobil oplevelse.

Det er dog min vurdering at man med et betalt mobil fokuseret UI bibliotek vil kunne få glæde af at bruge Next JS på den lange bane, da siden også skal agere salgskanal for produktet og derfor vil nyde godt af den bedre SEO samt generelle performance fordele.

Der er mange muligheder for at forbedre brugergrænsefladen, Det er min overbevisning at der skal lægges stor vægt interactions designet i en kommende beta test, for at finde de

steder hvor forbedringer kan blive gjort. Fra mit synspunkt er det datovælger og kalender oversigten som især kunne forbedres.

Spinoff har gennem projekt, med god grund været meget fokuseret på udseendet af brugergrænsefladen, hvilket der er brugt megen tid på, det betyder også, at der er lagt væsentligt mere tid i frontend udviklingen end backend udviklingen. Hvilket betyder at backenden i højere grad kunne bruge en refaktorering af kodebasen.

Jeg mener at Spinoff med denne prototype, med ganske få tilføjelser har god mulighed for at teste deres business case med dette produkt.

Derfor mener jeg også projektet har nået en tilfredsstillende konklusion med et produkt som imødekommer de problemer Spinoff har mede de eksisterende løsninger, hvilket Spinoff ligeledes har fundet tilfældet.

Bilag 1 - Oversigt over brugsscenarier

U1	Som anonym bruger skal jeg kunne afgive feedback efter et møde.
U2	Som bruger skal jeg kunne opret en aktivitet/event.
U3	Som bruger skal jeg kunne logge ind på systemet.
U4	Som virksomhedsadministrator eller administrator skal jeg kunne oprette spørgsmåls sæts.
U5	Som mødeleder skal jeg kunne slette mine egne eksisterende møder
U6	Som bruger skal jeg kunne se grafisk repræsenteret historisk feedback data vedrørende mine egne møder.
U7	Som bruger skal jeg kunne gennemse alle aktiviteter/events.
U8	Som ikke registreret bruger skal jeg kunne oprette en konto.
U9	Som bruger skal jeg kunne logge ud af systemet.
U10	Som bruger skal jeg kunne ændre login oplysninger.
U11	Som virksomhedsadministrator skal jeg kunne slette min virksomheds spørgsmål sæts.
U12	Som facilitator skal jeg kunne se feedback for enkelt møde.
U13	Som facilitator skal jeg kunne søge efter møder.
U14	Som bruger skal jeg kunne ændre mine personlige data
U15	Som bruger skal jeg kunne deaktivere min bruger
U16	Som virksomhedsadministrator eller administrator skal acceptere anmodninger om at tilhøre den pågældende virksomhed.
U17	Som Virksomhedsadministrator skal jeg kunne logge ind og se alle data fra de møde som er tilknyttet min virksomhed
U19	Som virksomhedsadministrator skal jeg kunne ændre min virksomheds spørgsmål sæts.
U21	Som mødeleder skal jeg kunne anmode om at blive en del af en virksomhed med et virksomheds ID
U6E	Som bruger skal jeg kunne exporter feedback data som en Excel fil
U23	Som mødeleder skal jeg kunne redigere mine egne eksisterende møder
U24	Som bruger skal jeg kunne modtage et reset link hvis jeg har glemt min kode.
U25	Som bruger skal jeg kunne konfirmere min bruger ved at modtage en mail med et unikt link.
U27	Som Mødeleder skal jeg kunne kategorisere mine begivenheder

Bilag 2 - Brugsscenarier og krav traceability matrix

Denne matrice viser hvilke krav som omfatter en eller flere use cases

Matricen indeholder kun funktionelle krav angivet i Kravspecifikation. Notationen er således RM1, for et krav som findes under “Must” og er nummer 1.

	RM1	RM2	RM3	RM4	RM5	RM6	RM7	RM8	RM9	RM10	RM11	RM12	RM13	RM14	RM15	RM16	RM17	RM18	RM19
U1					x														x
U2	x																		
U3				x															
U4																		x	
U5		x																	
U6					x	x									x		x		
U7							x	x			x					x			
U8												x	x						
U9				x															
U10										x									
U11																		x	
U12						x								x					
U16									x										
U19																		x	
U23			x																

De primære use cases består af U1 og U12. Disse to brugsscenarier udgøre kernen af den funktionalitet som systemet skal implementere.

Bilag 3 - JWT data

```
# HEADER: ALGORITHM & TOKEN TYPE
{
 "alg": "HS256",
 "typ": "JWT"
}

# PAYLOAD: DATA
{
 "sub": "be496b21-3e5f-458b-aaf7-11713dd932fe", // bruger id
 "CID": "1", // virksomheds id
 "CCON": "True", // bruger er bekræftet ansat af virksomheden
 "unique name": "admin@spinoff.com", // unik email
 "role": "Admin", // alle roller brugeren har
 "nbf": 1587971382,
 "exp": 1588576182, // udløbs tidspunkt
 "iat": 1587971382
}
```

Bilag 4 - Link til repositories

PWA: https://github.com/trolund/Feedback-PWA
Backend: https://github.com/trolund/Feedback-Backendv2

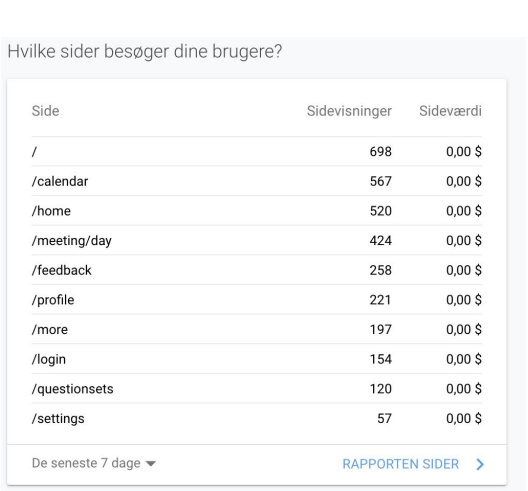
Bilag 5 - Deployed test version

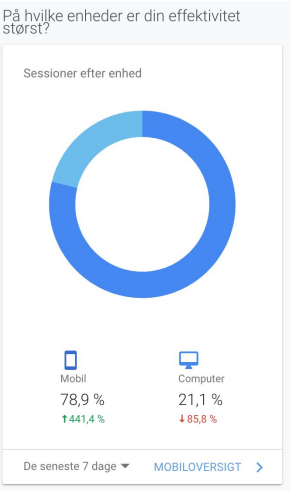
Frontend: https://feedbackspinoff.azurewebsites.net/

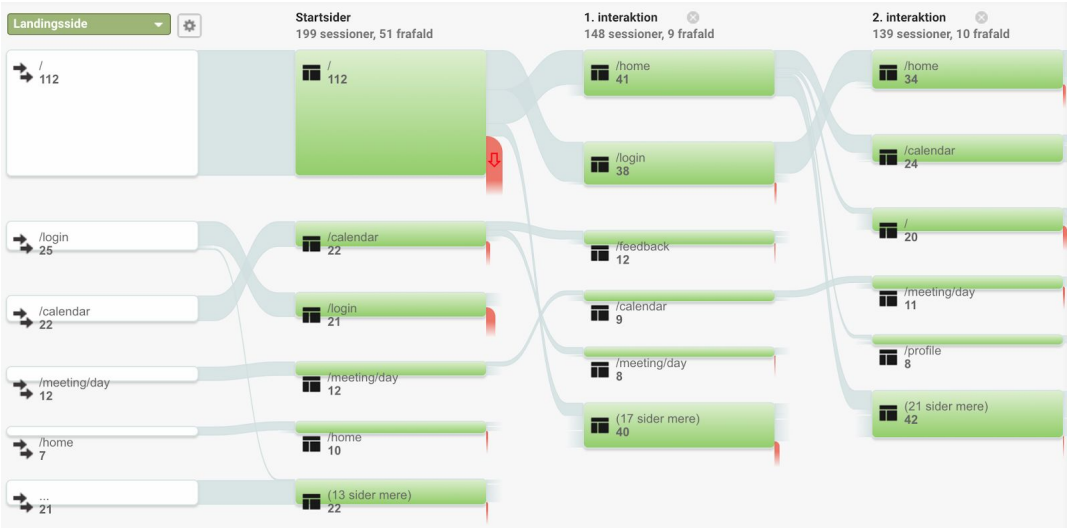
Backend: https://feedbackbackend.azurewebsites.net/

Bilag 6 - Nøgletal og grafer fra google analytics

Data er fra den 25 juni.







	Hændelseskategori	Al aktivitet	% Al aktivitet
1.	Launched	327	25,00 %
2.	Feedback Side	211	16,13 %
3.	Feedback-Question-Fetch	207	15,83 %
4.	Login	117	8,94 %
5.	Realtime Feedback Default	112	8,56 %
6.	Fingerprint	105	8,03 %
7.	Meeting Open	71	5,43 %
8.	Feedback	54	4,13 %
9.	Hide Templete Sets	33	2,52 %
10.	Animation	26	1,99 %

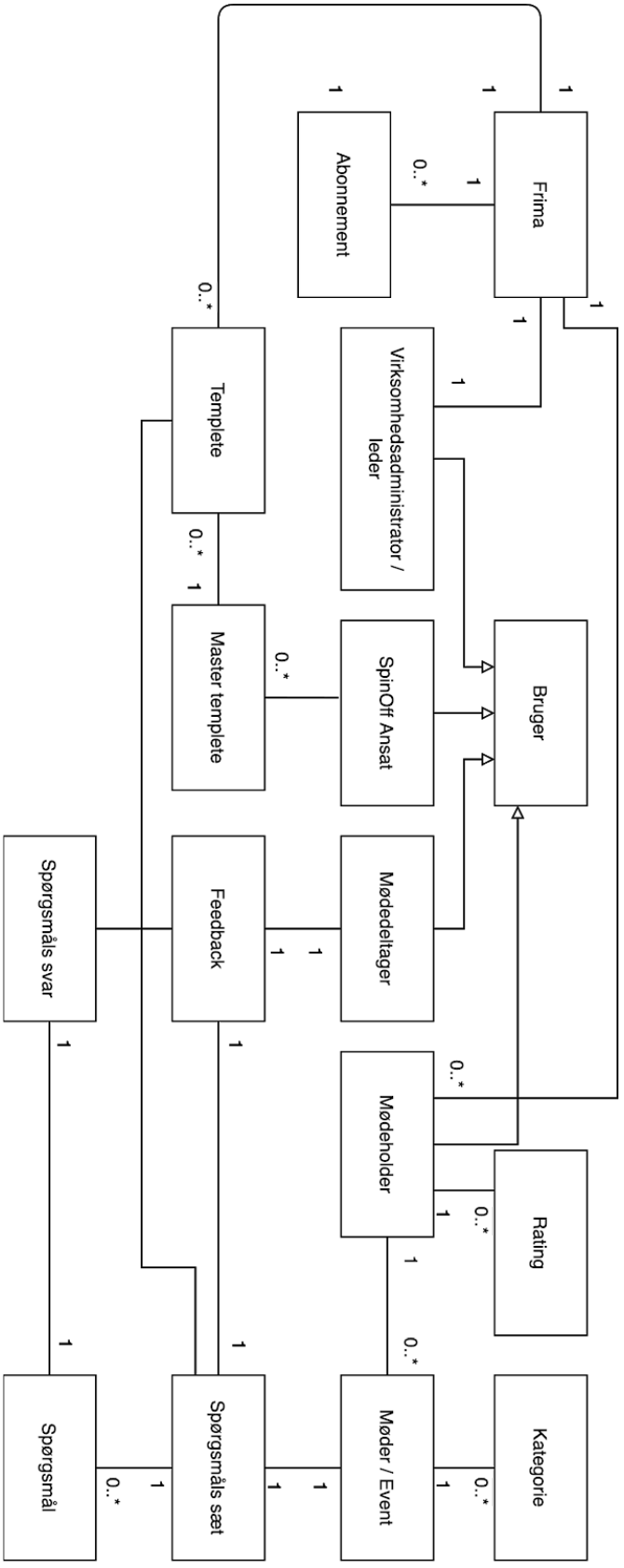
Bilag 7 - Testscenarier og brugsscenarie matrix

Udsnit af brugsscenarier og test cases matrix. Ud fra matricen kan det aflæses hvilke krav som er omfattet af et givent test case og vice versa.

Traceability matrix

	TC1	TC2	TC3	TC4	TC5	TC6
U1				x		
U2			x			
U3		x				
U4						x
U5					x	
U6					x	
U7				x		
U8	x					
U11						x
U12				x		
U19						x

Bilag 8 - Domæne model



Bilag 9 - Anden bruger test

Funktion	Beskrivelse	Testperson
	Rolle: Virksomhedsadm og mødeleder	Iphone 6
Startsiden	Knappen ”login” rettes til ”Login/Ny bruger”	Marie
Oprettelse af bruger (mobil)	Kasserne ”Eksisterende virksomhed” og ”Ny virksomhed” skal være lige store	Marie
Feedback, Konto confirmation email	- Afsender rettes til ”Opino” - Subject rettes det til ”Bekrætelse din konto” - Teksten i mailen rettes fra ”Confirm link: ….” til ”Bekræft din konto ved at trykke på dette link:…” - Selve linket skal være et hyperlink med en simplificeret tekst	Marie
Feedback, Ny bruger email	- Afsender rettes til ”Opino” - Teksten i mailen skal omformuleres	Marie
App navnet	- App ikonet på mobil skal ændres til Opino - Ligeledes i ens browservindue	Marie
Login	- Efter oprettelse af virksomhed og bekræftelse af bruger, så kunne jeg ikke logge ind på appen. Den sagde mit login var ugyldigt og jeg skal kontakte adm. Da jeg loggede på som admin kunne man se jeg var oprettet korrekt. Jeg måtte derfor trykke på ”glemt password” for at nulstille.	Marie
Feedback, Password reset – email	- Afsender rettes til ”Opino” - Simplificer link.	Marie
Feedback, Password	- Afsender rettes til ”Opino” - ”Confirmed” oversættes til ”bekræftet” - Tekst – ok J	Marie

confirmed – email		
Oversigt – home	- ”Personling score:” rettes til ”Personlig ….:” - Der er allerede 5 sorte stjerner ved min personlig score, men jeg har ikke oprettet eller fået feedback endnu. Så skal de ikke være grå? Eller betyder sort at de er tomme? - Kasserne ”Møder”, ”Undervisning”, ”Foredrag” – kan de være lige store? Men hvis det er på mobil versionen, så kan jeg godt se der ikke er meget plads til det… - ”Bygger på…..” rettes til ”?????”	Marie
Kalender, Start og slut tid	- Man skal trykke på urskiven for at vælge tidspunkt. Kan man lave hele feltet aktivt så man også kan trykke på selve tidspunktet? - Som default står begge tidspunkter som det samme tidspunkt. Hvis man gerne vil STARTE med at ændre STARTtidspunktet (som er mest intuitivt for mig) så kan jeg ikke indsætte et tidspunkt senere end slut tidspunkt. Da tidspunkterne står som det samme tidspunkt som default, og det betyder at man skal starte med at indsætte SLUT tidspunktet først.	Marie
1	- Når jeg opretter et møde, så indsætter det mødet til dagen efter. Mit møde er oprettet som den 26. maj, men i kalenderen er den under den 27. maj. Men inde mødet står det som den 26. maj. - Man kan kun oprette et møde hvis man indsætter et ”Navn på mødet” og vælger en ”Kategori”. Kan man gøre dette tydeligere, at det SKAL man, ellers oprettes mødet ikke? Eller er det overhovedet nødvendigt med en kategori, da man har valgt spørgsmålssæt og det siger vel hvilken kategori det er?	Marie

Møde – ændrer dato	- Når jeg ændrer datoen på et eksisterende møde, så kan jeg godt gemme mine ændringer, men de effektureres ikke, dvs mødet flyttes ikke til den nye dato i kalenderen. - MEN, man kan godt give feedback til mødet selvom det er den forkerte dato J	Marie
Kalender, Kategori	- Kan rækkefølgen af valgmuligheder være den samme som den første dropdown, hvor man skal vælge spørgsmålsæt? - Måske ”Kategori” skal helt væk, da det vil være den samme som valget af spørgsmålsæt? - Skærmen fryser når jeg ”venter” for længe og ikke er aktiv. - Når jeg vælger en kategori, kan man ikke se mit valg når jeg trykker på ”x” for at lukke vinduet.	Marie
Bruger – opdater bruger	- Hvis jeg trykker på ”Opdater bruger” og ændrer noget, så er der ikke en ”gem” knap el. lign. Så man ved ikke om den gemmer. Der er gem ved de andre funktioner, derfor kigger man også efter en ”gem” knap her.	Marie
Bruger – Skift kodeord	- Når jeg trykker på knappen, så kan jeg ikke skifte kodeord. Den ændrer kun farve til mørkegrøn.	Marie
Møde – ID	- Det er ikke så tydeligt, kan farven gøres mørkere?	Marie
Appen fryser	- Når den har været åben i ca. 30 min.	Marie
Efter mødefeedback		
Oversigt – home	- Jeg ved der er kommet 1 besvarelse tilbage. Men kan ikke se det på grafer eller som lagkage. Der står også der er 0 tilbagemeldinger. - Jeg kan se at ”Personlig score” har ændret sig fra 6 sorte stjerner til 3 grønne og 3 sorte, så det må betyde den har registreret 1 besvarelse.	Marie

Oversigt – Møder i uge 21	- Man kan kun se Uge 21, men det møde jeg har oprettet og modtaget feedback er søndag i uge 20. (i dag er søndag) så bør mødet kunne ses her?	Marie
Oversigt – Feedback fordelt på spørgsmål	- Her kan jeg ikke se nogle resultater da møde feedback ikke er registreret.	Marie
	Rolle: mødedeltager	iPad
QR kode	- ok	Kasper
Feedback - kommentar	- Kommentar feltet er centreret (irriterende) – kan det venstrestilles. - Der står ”Kommentar” som overskrift i feltet, ændres til ”Skriv dine kommentar her”, da det vil være mere opfordrende.	Kasper
Feedback – til sidst	- Når der blev trykket på ”send besvarelse” så sagde den ”Der skete desværre en fejl” og stort kryds, også kommer man tilbage til forsiden af appen. - Fejlmeddelelsen forsvinder for hurtigt – må gerne blive lidt længere på skærmen.	Kasper
Undervisnings-spørgesæt	- Hvordan vurderer du overordnet **set**…. – indsæt ”set” - Hvor tryg følte du dig …..- skal ændres til datid - Overvej direkte spørgsmål: o Hvordan var din egen indsats i undervisningen? o Hvor relevant var indholdet? o Hvordan var variationen i undervisningen? o Hvordan var undervisningen overordnet set?	Kasper

	-	
Tak for din besvarelse	- Må gerne blive lidt længere tid, forsvinder for hurtigt,. - Udnyt eksponeringen til noget reklame for Spinoff. (ex. SurveyMonkey gør det)	Kasper
	Rolle: Mødeleder (låner Maries mobil)	Maries mobil
Oversigt – vis/skjul knappen	- Ved diagrammet, stå de 3 kategorier, kan de sættes på 1 linje eller under hinanden med mindre bars, ikke centreret.	Kasper
Kalender	- Kalender knappen symboliserer en kalender. - Knap skal symbolisere det er her man opretter ny tilbagemelding. - Vigtigste funktionalitet på siden, er det lille + i højre hjørne. - Overskriften på siden er ”kalender” så det er ikke tydeligt det er her man opretter et møde. - + bør være større på siden, og når man har valgt det, så kan kalenderen evt. dukke frem, hvis der er behov for det. - Det er ikke kalenderen i sig selv der har værdi. Hvis man skal se hvilke møder man har kan man se i ”oversigt, fremtidige møder”.	Kasper

Kalender – opret nyt møde	- Farven fra kalenderen bevæger sig bagved pop-up vinduet, det er forvirrende. Og man kan rykke det ved at trykke ude på siden. Kan dette fjernes? - Datoen – kan ikke ændres. Det er nok fordi man valgte datoen til at starte med, men hvad nu hvis man har valgt forkerte dato. Kalender kan evt. komme op som en mulighed når man skal vælge dato for mødet. IKKE intuitivt at man skal trykke på kalender ikonet for at ændre dato. - Tidspunkt – Kasper trykker på uret intuitivt. Hele knappen skal man kunne trykke op. - Tidspunkt – når man har valgt sit tidspunkt står der nederst ”luk” og ”gem” – hvad er forskellen? - Når man har oprettet sit møde er ”Mødet er oprettet” og en bar der tæller ned og et x knap. Men den er der så kort tid at man tænker ”hvad er det? Er det en fortrydelsesbar?” men man kan ikke nå at reagerer hvis man skal trykke på noget. Så måske der bare skal stå ”mødet er oprettet” uden bar der tæller ned.	Kasper

Oprettet møde ”Test Kasper”	- Felterne står meget tæt på hinanden. - Møde ID kan man næsten ikke læse. - Dato, Start og slut står centreret – evt. dato på egen linje og start og slut på ny samme linje. - Man kan ikke se sit ”Katergori” valg. Tilbagemeldingssektionen: - ”Vis tilbagemelding løbende” – kan det stå på samme linje? - Trykker på skærm ikon, også står der ”Hvis feedback løbende ” – skal stå ens med det andet. Enten ”Feedback” eller ”tilbagemeldinger ”. - Hvad er forskellen når man trykker på ”vis tilbagemeldinger løbende” og når man vælger det samme efter skærmikonet? - De to faneblade nederst, er for tæt på linjen over. Så noget med afstand. - Hvis den store tykke grønne streg skal signalere nyt afsnit, så bør stregen være tyk eller ens ovenover ”Tilbagemelding” overskriften. - Antal besvarelser: vi skal blive enige om hvilket ord vi bruger ”Feedback, tilbagemelding, besvarelser”….. - Samlet resultat – skal være venstrestillet og med streg ovenover ligesom ved ”Tilbagemelding” overskriften. - ”Slet” knappen, skal flyttes til efter QR koden, så den er aller nederst. Overvej at bruge farven rød, for at signalerer at det er noget man virkelig skal overveje. - QR- koden: i teksten står der at man skal kunne kopiere koden ind i et slideshow, men når man trykker på koden med sin finger, så kan man ikke kopiere…. - Gem knappen – bar der tæller ned at ændringer er gemt….evt. slet baren der tæller ned. - Da vi ændrede datoen og gemte, så står den nye dato ikke på mødet.	Kasper

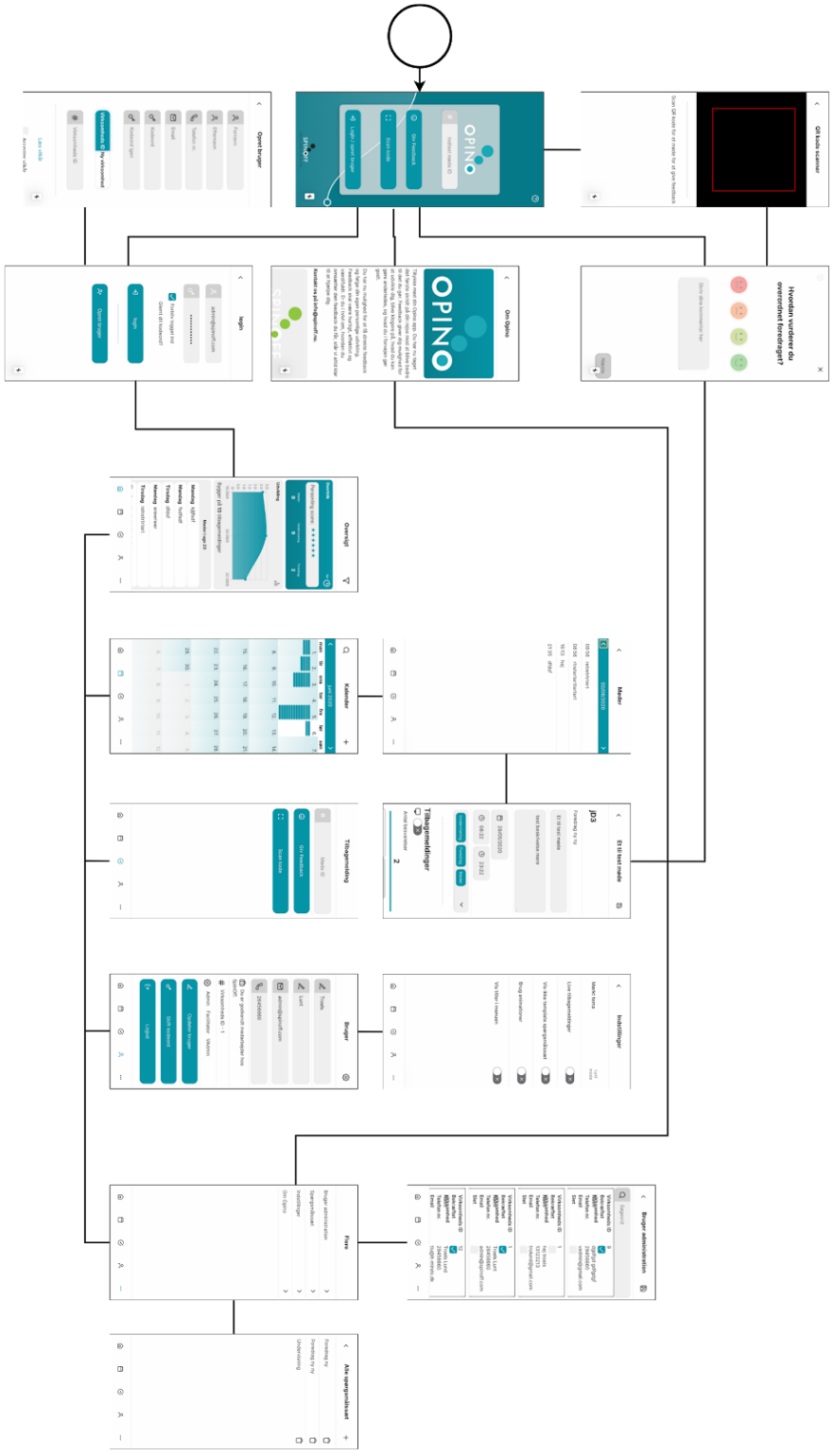
	- Husker man at gemme hvis man ændrer? Evt. En pop up om man ønsker at gemme hvis der sker ændringer på siden. Også kan Gem knappen fjernes.	
Bruger	- Hvis man skal ændre og gemme, så skal man trykke på en stor knap ”Opdater bruger”. På de andre sider er der en ”gem-ikon”, så det er ikke ens bygget op. Overvej at slette ”gem-ikonet”.	Kasper

…	- Om Opino: Email adressen skal man enten kunne trykke på hvis man vil skrive med det samme (email linke), eller trykke på for at kopiere.
	- Indstillinger – man kan tilgå dette sted 2 steder. På denne side og under ”Bruger, tandhjulsikonet”. Overvej at man kun kan tilgå det under ”…”
	- Overvej ”Bruger-ikonet” nederst ændres til ”Tandhjulsikonet” og her kan man se:
	  o Bruger profil
	  o Bruger adm
	  o Spørgsmålsssæt
	  o indstillinger
	- De tre prikker: her er der kun om Opino og kontaktinfo.
		Kasper

	Rolle: Elev	Iphone XR
	Det virkede fint.	Klara

Bilag 10 - App flow

I Screen Flow diagrammet indgår alle de side som en regular bruger vil se. Læg mærke til at nogle af siderne kan have et væsentligt andet udseende på baggrund af det stadige siden befinder sig i.



Bilag 11 - Oprindelig Kravspecifikation

Denne Kravspecifikation blev udarbejdet i tæt dialog med projektstiller ved opgavens start.

Functionality

Must

1. Systemet skal kunne oprette møder
2. Systemet skal kunne starte møder
3. Systemet skal kunne slette møder
4. Systemet skal kunne rette informationer om et existence møde.
5. Systemet skal indeholde et login system med 3 brugerroller
    a. Mødeleder/Underviser
    b. Virksomhedsadministrator
    c. Spinoff-administrator
6. Systemet skal have Minimum 2 externe authentication providers
    a. Google
    b. Microsoft
7. Systemet skal kunne gemme historiske data for afholdte møder
8. Systemet skal være i stand til at visualisere akkumuleret historisk data.
9. Systemet skal kunne vise alle oprettet aktiviteter for en given burger.
10. Systemet skal kunne vise en Virksomhedsadministrator alle aktiviteter i en virksomhed.
11. Systemet skal gøre det muligt for en bruger at tilhøre en virksomhed.
12. Systemet skal gøre brugeren i stand til at ændre sine egne login oplysninger.
13. Systemet skal kunne vise møderne i en Gregorian calendar.
14. Systemet skal kunne holde personlige data på en sikker både og i henhold til GDPR.
15. Systemet skal kunne vise en GDPR besked ved oprettelse af en bruger.
16. Systemet skal kunne visualisere feedback for et enkelt møde/event.
17. Systemet skal kunne visualisere at et møde er påbegyndt, afsluttet eller modtager feedback.
18. Systemet skal være i stand til at åbne og lukke feedback modtagelse.
19. Systemet skal kunne søge efter møde på navn og emne.
20. Systemet skal kunne vise alle kommende møder og afholdte møder som en liste.

Should

1. Systemet burde indeholde mulighed for at virksomheder kan betaling for tjenesten med tre forskelige type af abonnementer
    a. Basic - Gratis (30 dage max 5 brugere/ reg uden kreditkort oplysninger.)
    b. Business - Betalt
    c. Business pro - Betalt premium.

2. Systemet burde kunne generere strategisk rapporter som Excel filer fra det historiske data.
3. Systemet burde definere nye spørgsmåls-sæt såfremt virksomheden er betalende bruger.
    a. Herunder bør det være muligt at lave spørgsmåls-sæt med uvilkårlig antal spørgsmål.
4. Systemet burde kunne filtrere visning af historisk data på mindst et niveau, niveauer kunne være:
    a. Hvem mødelederen er
    b. Hvilke type/emne mødet har
    c.
5. Systemet burde være i stand til at validere feedback data således at kun "korrekt" data ender i systemet.
6. Systemet burde kunne automatisk stoppe feedback for et møde efter et tidsinterval efter slut tidspunktet som Virksomhedsadministratoren har fast sat, kunne fx være 2 timer efter mødet er slut.
7. Systemet burde kunne have mulighed for at vælge sprog (Liste i prioriteret rækkefølge).
    a. DK *
    b. ENG *
    c. NOR
    d. SWE
    e. ES
    f. Kinesisk

Could

1. Systemet kan have en "White label" løsning som lader kunder customize brugergrænsefladen til deres egen design guidelines. Her under:
    a. Logo
    b. Farver
    c. Typografi
    d. Evt. special login på subdomæne.
2.
3. Systemet kunne have mulighed for at tilpasse feedback-tidsintervallet (hvor lang tid efter deltagere kan give feedback efter mødet)
4. Systemet kunne være i stand til at "pushe" aktiviteter til brugerens online kalender, samt modtage aktiviteter fra samme online kalender. Dette kunne være (I prioriteret rækkefølge):
    a. Microsoft Outlook calendar
    b. Google calendar
5. Systemet kunne have mulighed for QR Koder til nemt at kunne give feedback på givent møde.
6. Systemet kunne have mulighed for at mødeledere live kan se deres feedback efterhånden som det bliver givet, herunder også antal deltagere som har besvaret.

7. Systemet Kunne have mulighed for at angive antal deltagere for et møde så det kan markeres som at have fuldendt feedback.
8. Systemet kunne være i stand til at sende mails ved konfirmering af kontooprettelse.
9. Systemet kunne være i stand til at sende mails med glemt kodeord.
10. Systemet kunne gøre Virksomhedsadministration i stand til at definere grupper inden for organisationen, fx. “Marketingafdelingen”, “Salg” eller “IT”.

Won’t

1. Systemet vil ikke kunne sende mails til alle mødedeltagere.
2. Systemet vil ikke være istand til at indeholde et invitationssystem/administrationssystem således at Virksomhedsadministrator kan acceptere eller afvise mødeledere for den givne virksomhed.
3. Systemet vil ikke have automatisk notifikation til mødeleder.’
4. Systemet kunne have automatisk notifikation / reminder til deltagere der mangler at besvare.
5. Systemet kunne gøre det muligt at definere lokationer så som “mødelokale 2, bygning 2”

Usability

1. Systemet skal kunne betjenes af ikke tekniske personale med lille til ingen support.
2. Systemet skal gøre det nemt for en deltagere at afgive feedback dvs. Under 3 tryk/klik for at kunne svare på spørgsmål.
3. Systemet burde kunne vise flotte animationer som underbygger hvad bruger ønsker.
4. Systemet skal være dokumenteret i en grad så det er muligt for en udefrakommende udvikler kan videreudvikle på projektet efter denne er læst såfremt udvikleren allerede har kendskab til de brugte teknologier.

Reliability

1. Systemet skal kunne håndtere conference størrelse store forsamlinger (2000-3000 deltagere) stabil vis.
2. Systems nede til skal være mindre end 2%.
3. Systemets møde opslag skal være sikkert nok til at det under 1/18000 gange er det korrekte møde som findes.

Performance

1. Systemet skal virke hurtige og responsivitet for bruger det vil sige der ikke må opleves længere ventetider end max 0.5sek.
2.
3. Systemet skal kunne være skalerbart til at kunne håndtere +1 million møder.

Supportability

1. Systemet burde kunne køre med minimal support fra Spinoff medarbejdere.

Design constraints

ingen

Implementation constraints

1. Udviklingen samt dokumentation skal kunne må foregå indenfor den aftalte projekt tid.

Interface constraints

1. Produktet bør kunne bruges både på mobile enheder såvel som en PC, men bør fokusere på mobile platforme.

Physical constraints

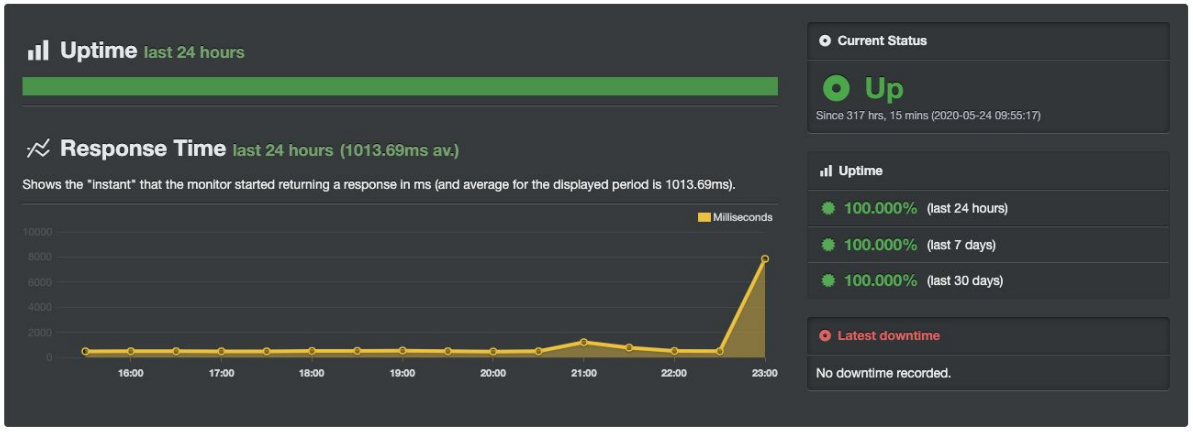
ingen

Bilag 12 - Uptime monitoreringsdata

Fra den 6 juni 2020.

Feedback backend

https://feedbackbackend.azurewebsites.net/health
A place to find all the details about your monitors

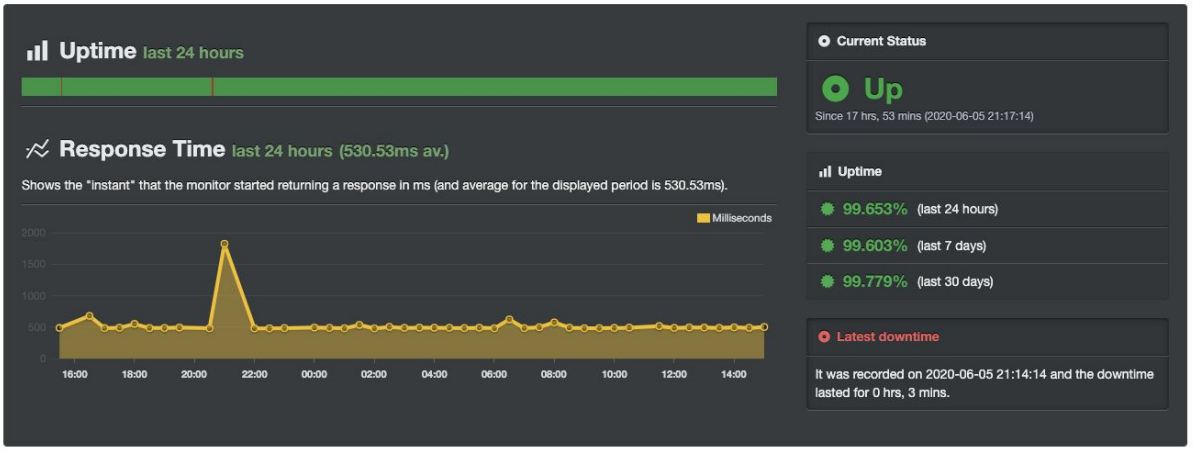


Latest Events (up, down, start, pause) Export Logs

Event	Date-Time	Reason	Duration
Up	2020-05-24 09:55:17	OK (200)	317 hrs, 15 mins
Started	2020-05-24 09:54:18	Started	0 hrs, 0 mins

feedbackspinoff PWA

https://feedbackspinoff.azurewebsites.net/
A place to find all the details about your monitors



Latest Events (up, down, start, pause) Export Logs

Event	Date-Time	Reason	Duration
Up	2020-06-05 21:17:14	OK (200)	17 hrs, 53 mins
Down	2020-06-05 21:14:14	Internal Server Error (500)	0 hrs, 3 mins
Up	2020-06-05 16:28:54	OK (200)	4 hrs, 45 mins
Down	2020-06-05 16:26:54	Internal Server Error (500)	0 hrs, 2 mins
Up	2020-06-04 06:15:30	OK (200)	34 hrs, 11 mins
Down	2020-06-04 06:08:30	Internal Server Error (500)	0 hrs, 7 mins

Bilag 13 - Tidsestimeringark

Tids estimeringen er vedlagt som excel fil:
Estimeringsark - komponentbaseret.xlsx

Bilag 14 - Milestoneplan

Milestoneplanen er vedlagt som pdf fil:
Milstoneplan.pdf

Bilag 15 - Project Scope Statement

Project Name: Applikation til optimering af møder og undervisning.
Projected start: 24/2

Completed by: 26/5 Projected duration: 18 uger

PROJECT PURPOSE:

Formålet med Feedback systemet er at hjælpe danske virksomheder, mødeledere og undervisere med at øge udbyttet af deres møder, undervisning eller lignende aktiviteter.

Dataerne kan akkumuleres for virksomheden eller organisationen så det også kan danne grundlag for strategiske indsatser og udviklingsområder.

PROJECT DESCRIPTION:

Udvikle et feedback system som kan som kan fungere på både Android, IOS samt i browseren.

DESIRED RESULTS:
Spinoff ønsker en løsning som de kan teste med eksisterende kunder. Denne løsning skal således kunne opsamle og visualisere data for kunden.

EXCLUSIONS: (per. 7/2)

- Invitations system til virksomhederne hvor de kan invitere mødeansvarlige.
- betalingsmodul
- Outlook integration

COMMUNICATION NEEDS:
Kommunikation vil ske som fysiske møder hver tredje uge. Kommunikation via mail vil ske efter behov.

ACCEPTANCE CRITERIA:
At systemet kan opsamle og vise data på en hensigtsmæssig måde.

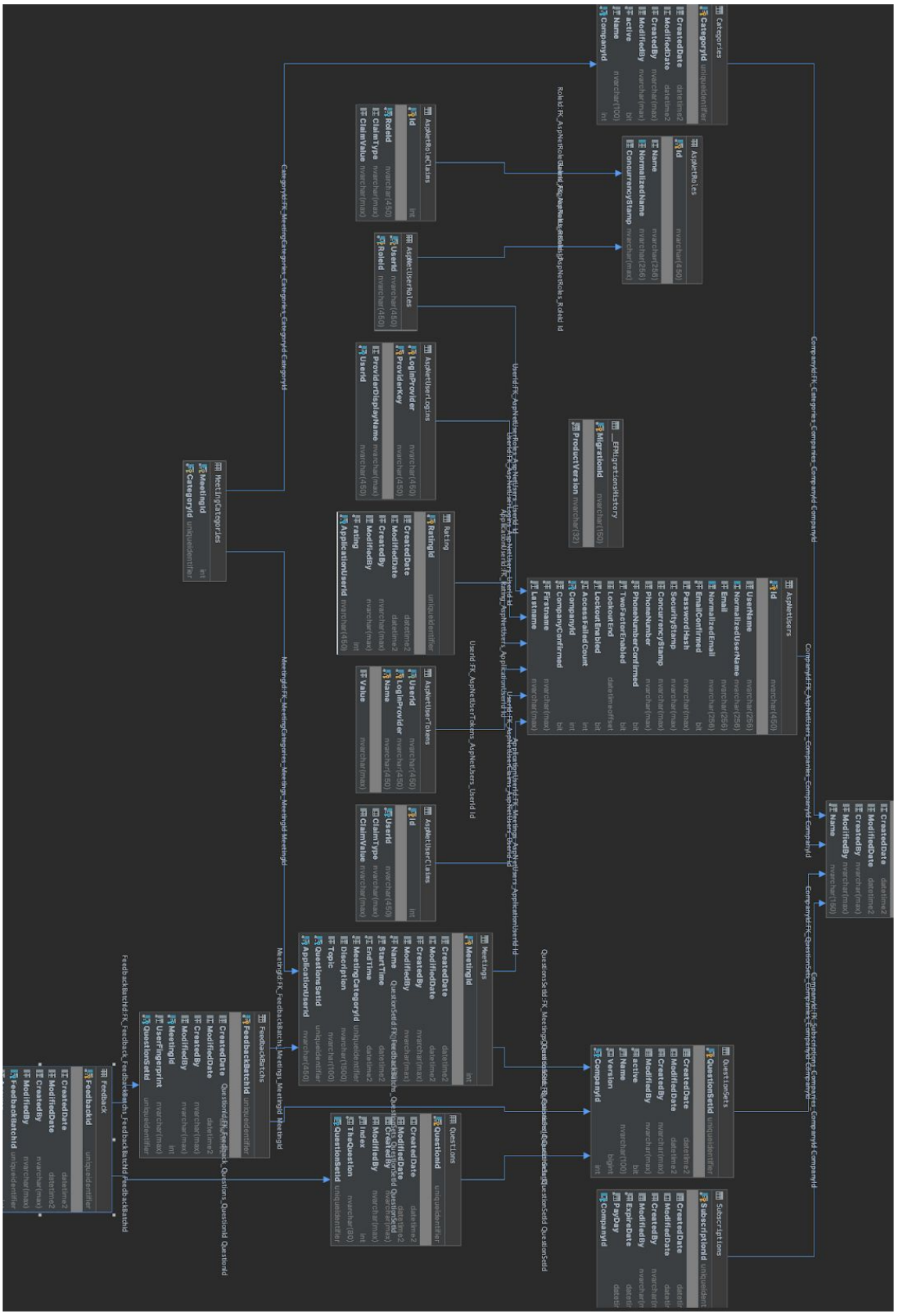
CONSTRAINTS:

APPROVALS (signatures) :

Person	**Signatur**
Tine Mørch Hansen	
Marie Rohrsted	
Troels Lund	

Bilag 16 - Database skemadiagram

Diagrammet er autogenereret med DataGrip



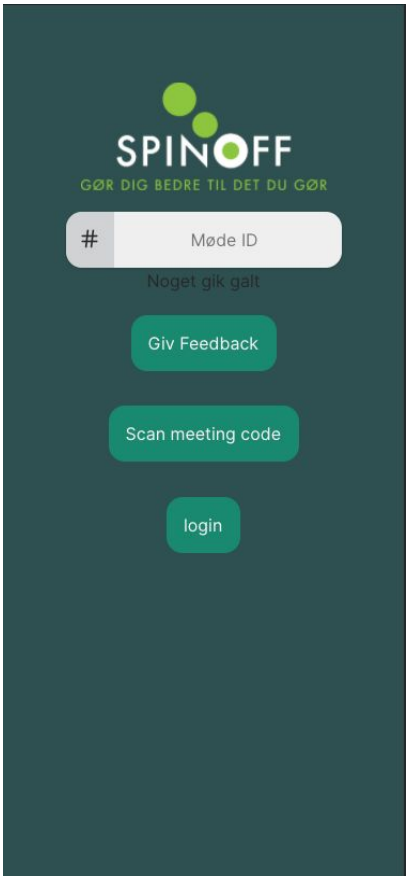
Bilag 17 - Første brugertest

Denne test er blevet udført med kunden, som også vil være slutbruger af produktet, dette betyder at nogle af de kommentarer, som kom ud af brugertesten er bundet op på rollen som kunde. Jeg har forsøgt at adskille dem i det nedenstående afsnit så det tydeligt kan ses, hvad jeg anser som et kundeønske og hvad der er et bruger ønske.

Generelt

Alt skal være på dansk.

Index side

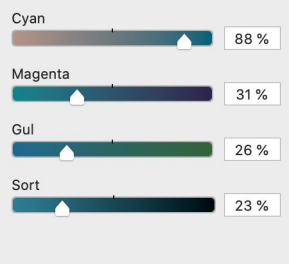


Brugerønsker

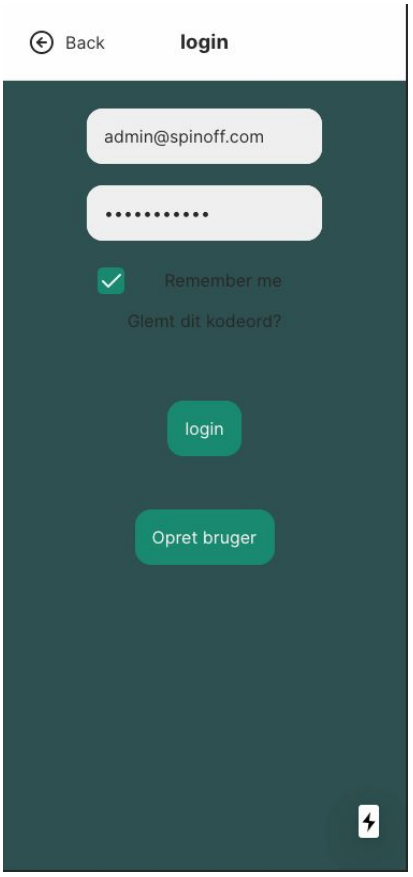
- Installations prompt besked bliver ikke vist på IOS for test brugerne.
- Brugerne ønsker at få knapperne til at have samme bredde og have samme afstand mellem sig.
- Brugerne synes ikke fejlmeddelelsen skal rykke knapperne.
- Derudover ønsker kunden at teamet (light/dark theme) kun kan indstilles af brugeren og derfor light som udgangspunkt.

Kunde

- Logo i toppen skal være “Opino logo” og så skal der stå powered by Spinoff i bunden.
- Knapfarven skal være den samme som farven på opino logo
- Baggrundsfarve:



Login side



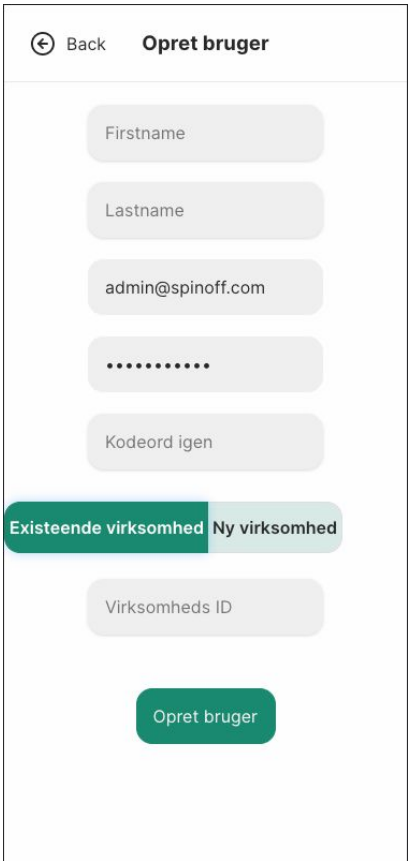
Bruger

- Det er svært at se remember me med den farve i forhold til baggrunden samt alt teksten være på dansk.

Kunde

- Baggrund skal være base farven og ikke den grønne farve.

Opret bruger side



Bruger

- Bruger det er svært som bruger at vide om der er fejl i inputtet da der ikke komme nogle fejlmeddelelse ved fejl input.

Kunde

- Toggle bør centerstilles (existerende/ny virksomhed)
- Kan knappen med eksisterende og ny virksomhed være samme størrelse som de andre felter

Oprettelse process

Kunden

- ønsker at bruger aktiverings siden skal være pænere og ikke blot tekst.

Dashboard/home side

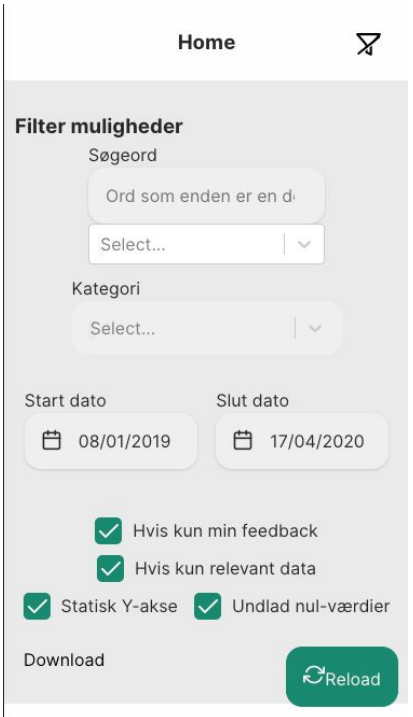
Bruger

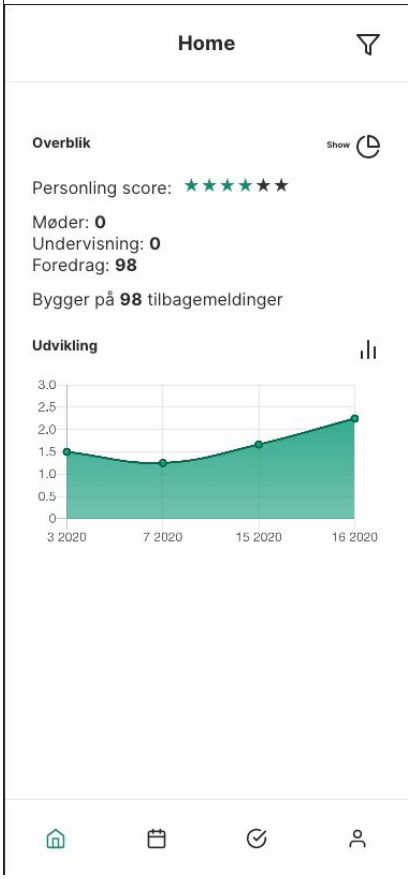
- Det er svært at gennemskue for brugeren hvad de forskellige filtre muligheder gør.

- Forslag til navne ændringer på checkbokse:
    - “Statisk Y-Akse” til “Hvis i forhold til Max”
    - “Undlad nul-værdier” er heller ikke særlig sigende men der var ikke umiddelbart et bedre bud. Evt. (Vis alle uger)

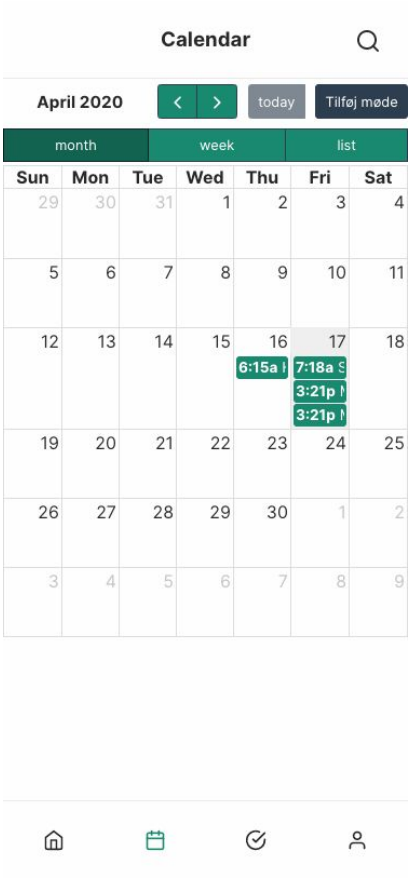
Kunde

- Ønsker ensartet input felter.
- Ønsker at alle filter muligheder bliver vist som en liste under hinanden.
- Ønsker en mulighed for at kunne se tilbagemeldinger fordelt på spørgsmål i et spørgsmåls set.
- Ønsker at den personlige score skal kun skal være baseret på de sidste 10 møder for en bruger.
- Ønsker at man kan se hvilke spørgsmål som bliver besvaret i Excel export.
- Ønsker ændring fra “select” til “vælg kategori”
- Ønsker venstrestilling af tekst





Kalender side



Bruger

- Ønsker en bedre både at se kalenderen på mobilenheder/små skærme.

Andre noter om kommende features:

Det er fint at alt feedback for et spørgsmåls sæt bliver slettet hvis et sæt opdateret.
Fingerprint for bruger der afgiver feedback

Bilag 18 - Brugsscenarie beskrivelser

Use case navn: Giv feedback til møde som anonym bruger
ID: U1
Scope: System under design process.
Level: Bruger mål.
Brief description: Mødedeltager skal kunne afgive feedback.
Primary actors: Mødedeltager
Secondary actors: Mødeleder
Stakeholders and interests: Mødedeltager, Mødeleder, Virksomhedsadministrator.
Precondition: Der er blevet: ● Oprettet et møde/event ● Mødet er blevet afholdt og er åbent for feedback
Main flow/ Main success Scenario: 1. Bruger tager mobil frem og åben app. 2. Bruger indtaster møde id. 3. Bruger trykker “Giv Feedback”. 4. Bruger svare på alle spørgsmål og trykker “Send Feedback”
Alternativ flow / Extensions: **Anvender QR-Kode:** 1. Bruger tager mobil frem og åben app. 2. Bruger trykker “Scan kode” 3. Bruger scanner QRkode. 4. Bruger bliver automatisk sendt til det rigtige møde når scanning er succesfuld. 5. Bruger svare på alle spørgsmål og trykker “Send Feedback”. **Anvender modtager link:** 1. Bruger åbner link i browser 2. Bruger svare på alle spørgsmål og trykker “Send Feedback”.
Special Requirements: Ingen
Postcondition: Mødet bliver lukket for Feedback.
Technology and data variations list: Mobil app / PWA

Frequency of Occurrence: Dette vil være den mest typiske use case i systemet.
Miscellaneous:

Use case navn: Opret møde
ID: U2
Scope: System under design process.
Level: Bruger mål.
Brief description: Mødeleder opretter aktivitet.
Primary actors: Mødeleder.
Secondary actors: Mødedeltager.
Stakeholders and interests: Mødedeltager, Mødeleder, Virksomhedsadministrator.
Precondition: Mødelederen er oprettet som bruger i systemet.
Main flow/ Main success Scenario: 1. Mødeleder logger ind. 2. Mødeleder trykker “Kalender” 3. Mødeleder trykker “Opret Møde/+” 4. Mødeleder får vist form med oplysningerne: a. Starttidspunkt og dato for møde b. Emne. c. Kort beskrivelse. d. Hvilke spørgsmåls set som skal bruges. 5. Mødeleder udfylder form og trykker Opret møde. 6. Mødelederen ser toast besked med mødet er korrekt oprettet.
Alternativ flow / Extensions: 1. Mødeleder logger ind. 2. Mødeleder trykker “Kalender” 3. Mødeleder trykker “Opret Møde/+” 4. Mødeleder får vist form med oplysningerne: a. Starttidspunkt og dato for møde b. Kort beskrivelse. c. Emne. d. Evt. mails på alle deltagere. 5. Mødeleder udfylder form og trykker Opret møde. 6. Mødelederen ser toast besked med mødet ikke korrekt oprettet og må gentage fra step 4 og får feedback på hvilke felter der ikke var korrekte.

Special Requirements:
Postcondition: Ingen
Technology and data variations list:
Frequency of Occurrence: Sker ved alle mødeoprettelser.
Miscellaneous:

Use case navn: Login
ID: U3
Scope: System under design process.
Level: Bruger mål.
Brief description: Mødeleder Logger ind på Webgrænsefladen
Primary actors: Mødeleder
Secondary actors: ingen
Stakeholders and interests: Virksomhedsadministrator.
Precondition: Mødeleder har en bruger og er tilknyttet en virksomhed, og er ikke logget ind i forvejen.
Main flow/ Main success Scenario: 1. Mødeleder åbner browser 2. Mødeleder skriver URL til systemet i top baren. 3. Mødeleder bliver mødt af startside. 4. Mødeleder klikker login. 5. Mødeleder skriver email og kodeord. 6. Mødeleder klikker login. 7. Mødeleder bliver sendt til sin oversigtsside.
Alternativ flow / Extensions: ingen
Special Requirements: External login providers (Google og Microsoft)
Postcondition: ingen.
Technology and data variations list:
Frequency of Occurrence: Ofte

Miscellaneous:

Use case navn: Opret spørgsmåls sæt
ID: U4
Scope: System under design process.
Level: Bruger mål.
Brief description: Som mødeleder eller Virksomhedsadministrator kan man oprette spørgsmåls sæt som kan bruges til hvert møde.
Primary actors: Virksomhedsadministrator
Secondary actors: Mødeleder eller Virksomhedsadministrator, Deltager/feedback giver
Stakeholders and interests: Mødeleder eller Virksomhedsadministrator, Deltager/feedback giver
Precondition: Mødeleder eller Virksomhedsadministrator er oprettet i systemet. Og er Logget ind.
Main flow/ Main success Scenario: 1. Burger klikker på “mere” 2. Brugeren klikker på “Spørgsmåls sæt” i listen på mere-siden hvilket bringer brugeren til en ny side. 3. Bruger klikker “Opret spørgsmåls sæt”/”+“ 4. Bruger bliver vist en liste med et element i som indeholder et tomt input felt. 5. Bruger udfylder input felt 6. Bruger klikker “Tilføj spørgsmål” nyt tomt element bliver tilføjet. 7. Brugeren udfylder element. (Trin 5 og 6 kan udføres flere gange for at lave flere spørgsmål. Et spørgsmåls sæt er et validt sæt sålænge det er minimum et spørgsmål). 8. Bruger klikker på “Opret Spørgsmåls sæt“. 9. Brugeren bliver sendt tilbage til side under “Spørgsmåls sæt” hvor det nye sæt nu bliver vist i en liste.
Alternativ flow / Extensions: Ingen
Special Requirements: Ingen
Postcondition: Ingen
Technology and data variations list:

Frequency of Occurrence:
Miscellaneous:

Use case navn: Slette exsiterende møder
ID: U0
Scope: System under design process.
Level: Bruger Mål
Brief description: Bruger ønsker at slette et møde som endnu ikke har modtaget feedback
Primary actors: Fatalicator
Secondary actors: Virksomhedsadministrator
Stakeholders and interests: Virksomhedsadministrator,Fatalicator
Precondition: Mødet er oprettet og har endnu ikke modtaget nogle tilbagemeldinger.
Main flow/ Main success Scenario: 1. Bruger logger ind 2. Bruger navigere til mødet via kalender 3. Bruger scroller til bunden og trykker slet 4. Bruger trykker bekræft i modal.
Alternativ flow / Extensions: ingen
Special Requirements:
Postcondition:
Technology and data variations list:
Frequency of Occurrence: relativt ofte.
Miscellaneous:

Use case navn: Gennem ser historisk data (Dashboard)

ID: U6

Scope: System under design process.

Level: Bruger mål.

Brief description: Virksomhedsadministrator ønsker at få indsigt i data omkring virksomhedens mødeledere.

Primary actors: Virksomhedsadministrator

Secondary actors: Mødelederen

Stakeholders and interests: Mødelederen, Virksomhedsadministrator

Precondition: Der er blevet afholdt en række møder med forskellige mødeleder som høre under den pågældende virksomhed hvor Virksomhedsadministrator høre til. Virksomhedsadministrator er logget ind og befinder sig på dashboardet.

Main flow/ Main success Scenario:

1. Bruger klikker på filter icon
2. Bruger indtaster alle de ønsket oplysninger i input felter
    a. Start og slut data
    b. Fordelt på uger eller måneder
    c. Evt. søge ord
    d. Evt kategorier
    e. Vis kun feedback
    f. Vis kun relevant data
    g. Statisk Y-akse
    h. Vis kun perioder med registreret aktiviteter

Efter dette burde alle relevante data som opfylder de indtastet oplysninger være tilstede vist som en graf i dashboardet.

Alternativ flow / Extensions:

Efter udførelsen af main flow kan de data som er flirtet downloads som en excel fil.

Dette gøres:

1. Klik “sky med pil ned”

Special Requirements:

Postcondition:

Technology and data variations list:

Frequency of Occurrence: Formodet at dette vil ske relativt sjælent.

Miscellaneous:

Use case navn: Gennemse alle aktiviteter.
ID: U7
Scope: System under design process.
Level: Bruger mål.
Brief description: Virksomhedsadministrator ønsker at få indsigt i data omkring virksomhedens mødeledere.
Primary actors: Virksomhedsadministrator
Secondary actors: Mødelederen
Stakeholders and interests: Mødelederen, Virksomhedsadministrator
Precondition: Der er blevet afholdt en række møder med forskellige mødeleder som høre under den pågældende virksomhed hvor Virksomhedsadministrator høre til. Virksomhedsadministrator er logget ind i webgrænsefladen.
Main flow/ Main success Scenario: 1. Burger klikker på eventkalender 2. Vælg båden at se events på a. Bruger klikker på Agenda (for at se liste over events) b. Bruger klikker på måned (for at se en hel måned) c. Bruger klikker på uge (for at se en uges events) 3. Uger/månder ændres med pilene eller ved swipe Brugeren bør kunne se alle events som er inden for det valgte interval.
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:
Frequency of Occurrence:
Miscellaneous:

Use case navn: Burger registering
ID: U8
Scope: System under design process.
Level: Bruger mål
Brief description: For en bruger kan oprette events og modtage feedback skal personen have en bruger i systemet.
Primary actors: uregistrert bruger
Secondary actors: ingen
Stakeholders and interests: Virksomhedsadministrator og administrator
Precondition: bruger har navigeret til roden af siden
Main flow/ Main success Scenario: 1. Trykker login/opret bruger 2. Trykker opret bruger 3. Indtaster oplysninger 4. Vælger eksisterende virksomhed 5. Indtaster id på virksomhed 6. Bekræfter samtykkeerklæring 7. Trykker opret
Alternativ flow / Extensions: **4 og 5 udskiftes til:** Vælger ny virksomhed Indtaster virksomhedsnavn
Special Requirements:
Postcondition:
Technology and data variations list:
Frequency of Occurrence: sjælent
Miscellaneous:

Use case navn: Bruger logud

ID: U9
Scope: System under design process.
Level: Burger mål
Brief description: Efter brugen er logget ind i systemet skal brugen kunne logge ud.
Primary actors: Mødelederen, Virksomhedsadministrator, Administrator (Burger oprettet i systemet)
Secondary actors:
Stakeholders and interests:
Precondition: Burgeren er logget ind i systemet
Main flow/ Main success Scenario: 1. Gå til profil side 2. Klik på logud
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:
Frequency of Occurrence:
Miscellaneous:

Use case navn: Bruger ændre login oplysninger
ID: U10
Scope: System under design process.
Level: Burger mål
Brief description: Brugeren ønsker at ændre password.
Primary actors: Bruger
Secondary actors: ingen
Stakeholders and interests: Kun den bruger

Precondition: bruger er på logget ind og er på profilsiden.
Main flow/ Main success Scenario: 1. Klik på Skift kodeord 2. Indtast nuværende kodeord og det ny ønsket kodeord 3. Bekræftelsesmail modtages.
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:
Frequency of Occurrence: sjælent
Miscellaneous:

Use case navn: Mødeholder ser feedback for enkelt møde
ID: U12
Scope: System under design process.
Level: Burger mål
Brief description: Bruger har modtaget feedback på et møde og ønsker at se denne feedback
Primary actors: Facilitator
Secondary actors: Virksomhedsadministrator
Stakeholders and interests: Virksomhedsadministrator og Facilitator
Precondition: Bruger har modtaget feedback på et møde og befinder sig på kalenderen-siden.
Main flow/ Main success Scenario: 1. Burger trykker på det møde som har modtaget feedback 2. Bruger scroller ned og ser tilbage meldinger 3. Bruger trykker på kommentare for at se kommentare tilknyttet spørgsmålene 4. Bruger ser sin overordnet score for mødet
Alternativ flow / Extensions:
Special Requirements:

Postcondition:
Technology and data variations list:
Frequency of Occurrence: ofte
Miscellaneous:

Use case navn: Søg efter gammelt møde
ID: U13
Scope: System under design process.
Level: Burger mål
Brief description: Bruger ønsker at søge efter møde
Primary actors: Administrator, Virksomhedsadministrator, Facilitator
Secondary actors: ingen
Stakeholders and interests: Administrator, Virksomhedsadministrator, Facilitator
Precondition: Der er blevet oprettet et møde som indeholder en frase i enden beskrivelsen eller navnet. Brugeren er logget ind og er på kalendersiden.
Main flow/ Main success Scenario: 1. Tryk på spøge-ikon → Et søgefelt kommer frem 2. en søge frase indstates i søgefelt → Kalenderen viser kun events som matcher søgefrasen.
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:
Frequency of Occurrence:
Miscellaneous:

Use case navn: Som bruger skal jeg kunne ændre mine personlige data

ID: U14
Scope: System under design process.
Level: Bruger mål
Brief description: Brugeren ønsker at ændre sine personlige data.
Primary actors: Alle brugere
Secondary actors:
Stakeholders and interests:
Precondition: Brugeren er logget ind og er navigeret til profilsiden.
Main flow/ Main success Scenario: 1. Bruger ændre de oplysninger som brugeren ønsker at ændre. 2. Bruger trykker på opdater bruger
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:

Use case navn: Som bruger skal jeg kunne deaktivere min bruger
ID: U15
Scope: System under design process.
Level: Bruger mål
Brief description: Bruger ønsker ikke længere at bruge systemet og vil derfor deaktivere sin bruger.
Primary actors: Alle brugere
Secondary actors: ingen
Stakeholders and interests: ingen
Precondition: Brugeren er logget ind og er på profilsiden.
Main flow/ Main success Scenario: 1. Burger trykker på “slet bruger” knap.

2. Bruger bliver sendt til login side.
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:

Use case navn: Som Virksomheds admin skal jeg kunne invitere mødeleder samt acceptere anmodninger om at tilhøre den pågældende virksomhed.
ID: U16
Scope: System under design process.
Level: Bruger mål
Brief description:
Primary actors: Virksomhedsadministrator
Secondary actors: Administrator
Stakeholders and interests: Facilitator
Precondition: Virksomhedsadministrator er logget ind og er på brugeradministration siden
Main flow/ Main success Scenario: 1. Brugeren som ønsker at blive en del af virksomheden kan ses i liste 2. Bruger trykker på checkbox under “Bekræftet virksomhed” og i rækken med den ønsket bruger. 3. Bruger trykker på gem.
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:

Use case navn: Som Virksomhedsadministrator skal jeg kunne logge ind og se alle data

fra de møde som er tilknyttet min virksomhed
ID: U17
Scope: System under design process.
Level: Bruger mål
Brief description: Virksomhedsadministrator er logget ind og er på kalendersiden
Primary actors: Virksomhedsadministrator
Secondary actors:
Stakeholders and interests:
Precondition:
Main flow/ Main success Scenario: Alle møder som er tilknyttet en bruger af samme virksomhed som virksomhedsadministration administrere.
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:

Use case navn: Som virksomhedsadministrator skal jeg kunne ændre min virksomheds spørgsmål sæts.
ID: U19
Scope: System under design process.
Level: Bruger mål
Brief description: virksomhedsadministrator ønsker at ændre virksomhedens spørgsmåls sæt
Primary actors: virksomhedsadministrator
Secondary actors: Facilitator
Stakeholders and interests:
Precondition: Virksomhedsadministratoren er logget ind og er på spørgsmåls sæt siden.

Main flow/ Main success Scenario: 1. Burger trykker på det spørgsmåls sæt som ændre der skal ændres 2. Bruger kan ændre a. Spørgsmålsformulering b. Spørgsmåls rækkefølge c. Spørgsmåls sæt navn 3. Der trykkes på gem
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:

Use case navn: Som mødeleder skal jeg kunne anmode om at blive en del af en virksomhed med et virksomheds ID
ID: U21
Scope: System under design process.
Level: Bruger mål
Brief description: Brugeren skal være en del af en virksomhed.
Primary actors: Facilitator
Secondary actors: Virksomhedsadministrator
Stakeholders and interests: Anmodningen sker blot ved at angive virksomheds id ved registrering
Precondition:
Main flow/ Main success Scenario:
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:

Use case navn: Som mødeleder skal jeg kunne redigere mine egne eksisterende møder

ID: U23
Scope: System under design process.
Level: Bruger mål
Brief description:
Primary actors: Facilitator
Secondary actors: ingen
Stakeholders and interests: Virksomhedsadministrator, Administrator
Precondition: Brugeren er logget ind og har oprettet et møde. brugeren er navigeret til Kalenderen.
Main flow/ Main success Scenario: 1. Burger trykker på møde i Kalender 2. Bruger redigere oplysninger om mødet 3. Bruger trykker gem
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:

Use case navn: Som bruger skal jeg kunne modtage et reset link hvis jeg har glemt min kode.
ID: U24
Scope: System under design process.
Level: Bruger mål
Brief description: Bruger har mistet sit password og ønsker et nyt password
Primary actors: Bruger
Secondary actors:
Stakeholders and interests:
Precondition: Bruger er navigeret til login side

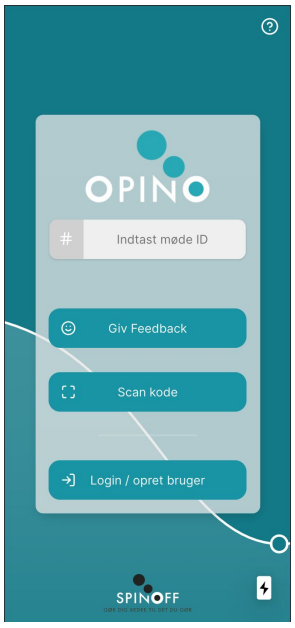
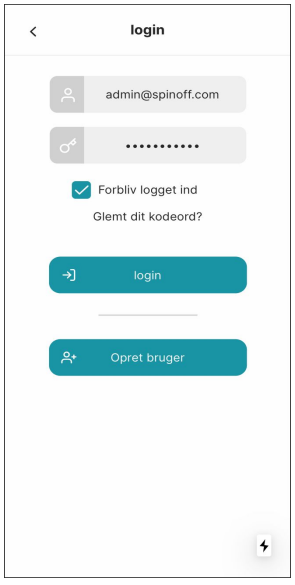
Main flow/ Main success Scenario: 1. bruger trykker på glemt password 2. bruger indtaster email tilknyttet til bruger 3. bruger trykker på “Send password reset forspørgsel” 4. bruger modtager en mail med et link som der trykkes på 5. bruger indtaster nyt kodeord to gange. 6. og trykker ok
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:

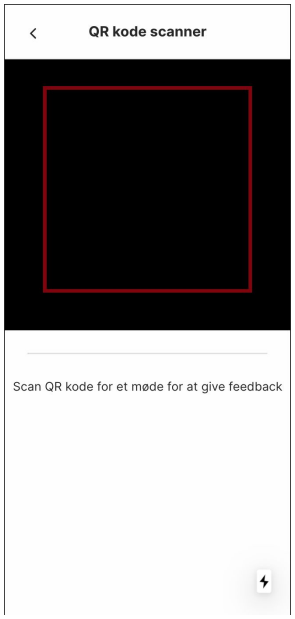
Use case navn: Som bruger skal jeg kunne konfirmere min bruger ved at modtage en mail med et unikt link.
ID: U25
Scope: System under design process.
Level: Bruger mål
Brief description: En bruger kan ikke logge ind før brugeren er blevet aktiveret via et link send til brugerens email.
Primary actors: Bruger
Secondary actors:
Stakeholders and interests:
Precondition: Burger har oprettet en konto og venter på email.
Main flow/ Main success Scenario: Burger klikker på link i e mail og er nu istand til at logge ind.
Alternativ flow / Extensions:
Special Requirements:
Postcondition:
Technology and data variations list:

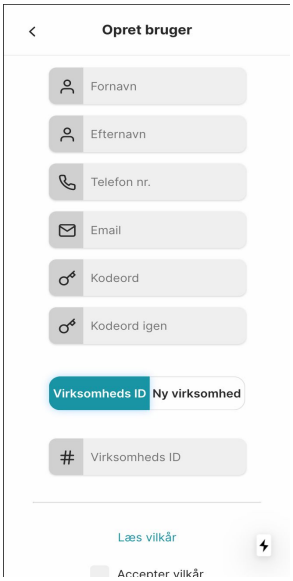
Use case navn: Som mødeleder skal jeg kunne kategorisere mine begivenheder
ID: U27
Scope: System under design process.
Level: Bruger mål
Brief description:
Primary actors: Facilitator
Secondary actors: Virksomhedsadministrator
Stakeholders and interests:
Precondition: Brugeren er logget ind og er på kalendersiden
Main flow/ Main success Scenario: 1. Tryk “+” 2. Tryk på kategori vælger 3. Vælg kategorier 4. Udfyld andre informationer 5. Tryk opret
Alternativ flow / Extensions: 1. Tryk på møde 2. Tryk på kategori vælger 3. Tilføj/slet kategori 4. Tryk gem
Special Requirements:
Postcondition:
Technology and data variations list:

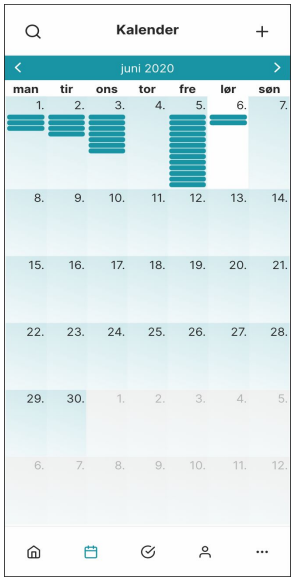
Bilag 19 - PWA sider

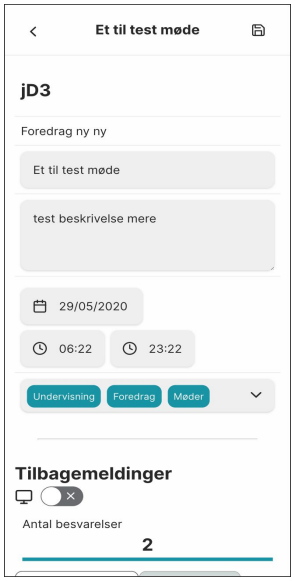
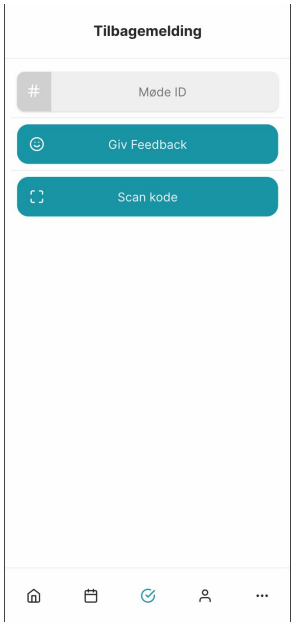
Overblik over sider som findes i PWA'en

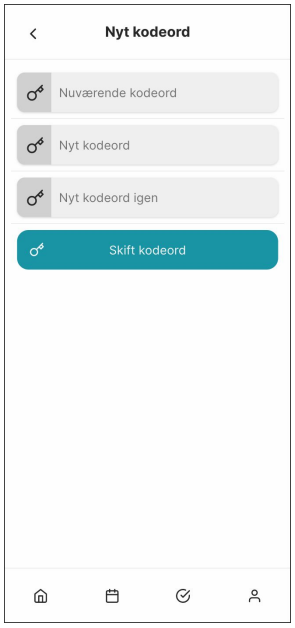
	**1** Index side (første side brugeren ser når du navigere til siden)
	**2** Login side

	**3** QR Kode scanner
	**4** Siden om app'en samt kontaktinformation

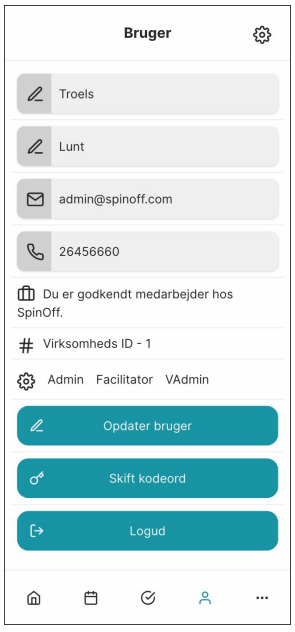
	**5** Brugeroprettelse
	**6** Siden går under flere navne i dokumentationen (Home/dashboard/oversigt)

	**7** Kalender bruges både til overblik over events og man kan oprette et møde ved at klikke +.
	**8** “En dags siden” viser en list i sorteret række følge over evnet på en given dag angivet med datoen i den blå bar.

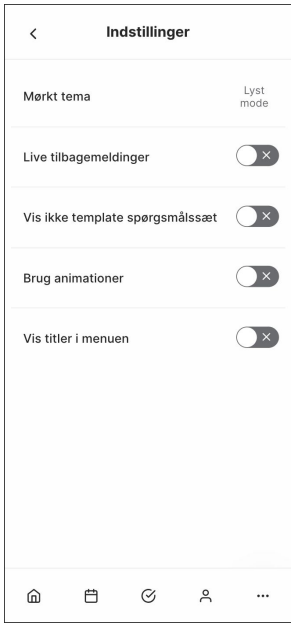
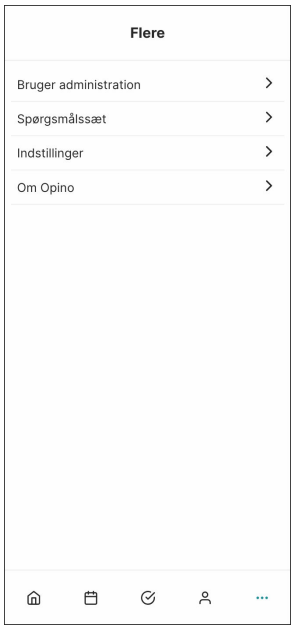
	**9** Side for et enkelt møde hvor det både er muligt at se og ændre møde oplysninger samt se feedback for kun dette specifikke møde
	**10** Denne side fungere ens med index siden men er for brugere som er logget ind i systemet og ønsker at give feedback til en anden facilitator.

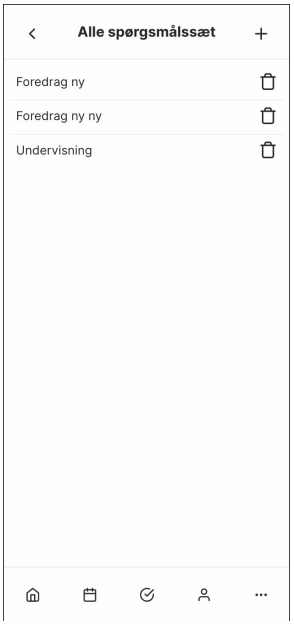
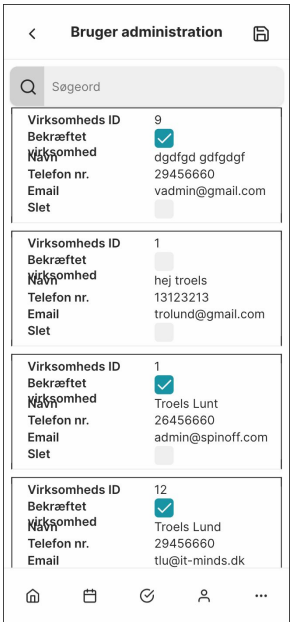


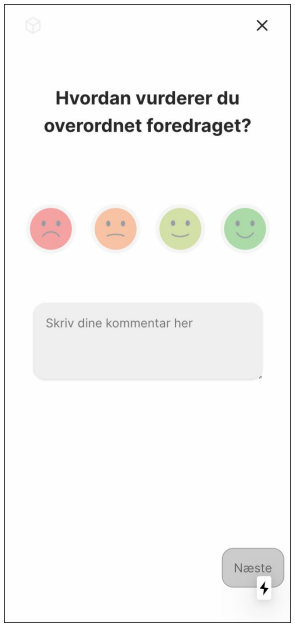
12
Anmod om et nyt kodeord.



13
Profil side med informationer om brugeren som er logget ind samt mulighed for at opdatere brugeroplysninger.

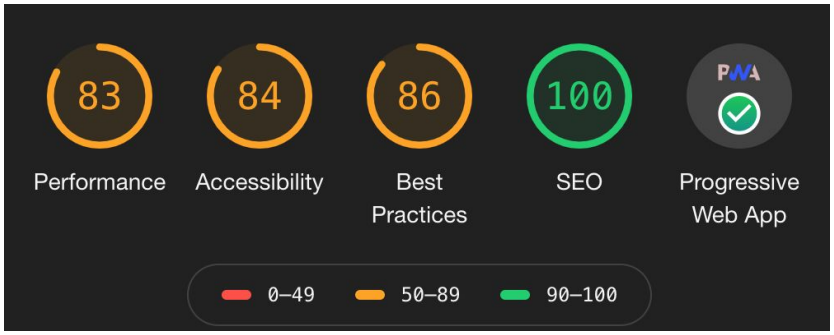
	**14** Indstilinger af appen.
	**15** Mere side bruges til at navigere til andre mindre brugte sider.

	**16** liste over alle spørgsmåls sæt brugeren har adgang til.
	**17** simple bruger administration

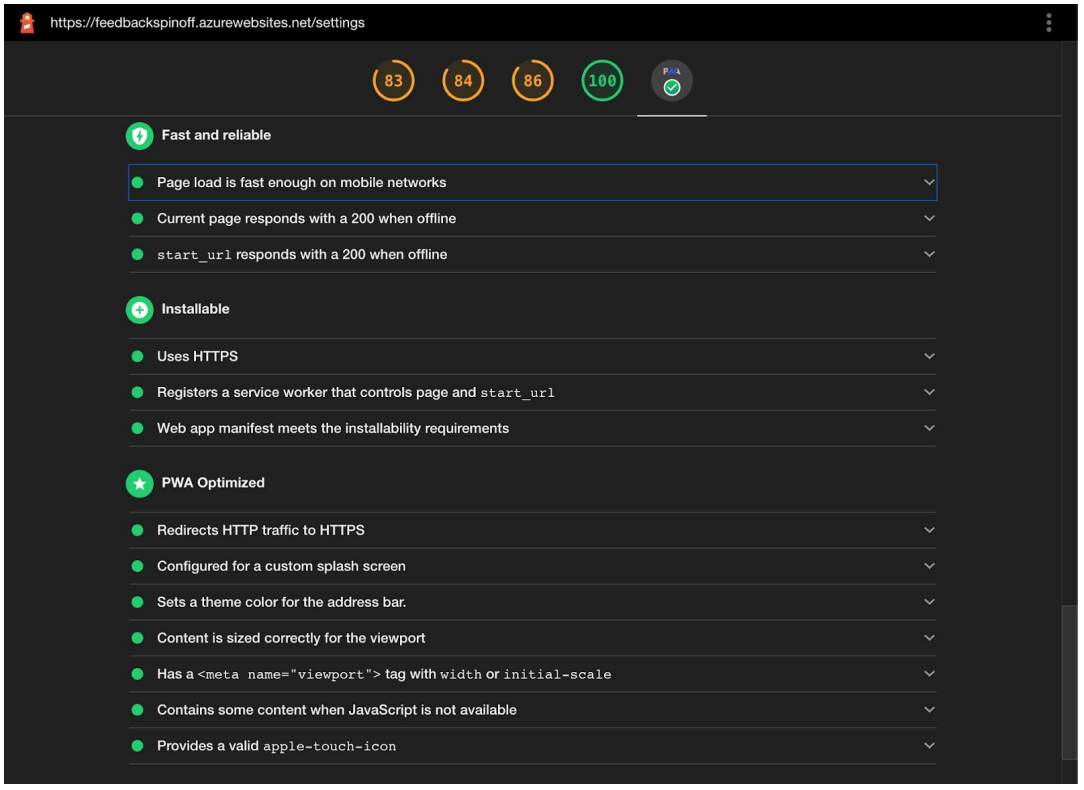
	**18** Side hvor Feedback afgives.

Bilag 20 - Lighthouse evaluering

Overordnet evaluering



PWA benchmark



Bilag 21 - Udviklingsmanual

Miljøvariable og konfiguration

Start kommando

Hver app service har et startup script, som aktiveres så snart en ny version af den kompilerede kode er hentet til maskinen. For frontenden er det ikke eksplicit, da det er standard at ved deployment af node Node baseret applikationer, at NPM run start køres, hvilket i dette tilfælde vil starte NextJS serven. Backend er det angivede start script *dotnet WebApi.dll,* Som tidligere beskrevet indeholder WebApi projektet entry pointet og det er derfor denne som kaldes.

Alle oplysninger som skal kunne ændres på baggrund af det miljø systemet opsættes på er angivet som miljøvariabler i backend projektet og kan derfor nemt ændres i Azure.

Frontend projektet har desværre vist sig besværligt at få next konfigurations filen til at bruge miljøvariable på azure og derfor er der her blevet lavet et workaround, som virker ved der findes to next konfigurationsfiler i roden af projektet henholdsvis *next.config.js* og *next.config.azure.js* i pipelinen erstattet *next.config.js* med azure versionen således at det er denne som bliver brugt i produktion og den anden lokalt i udvikling. Dette er ikke en optimal løsning, men gør stadig at deployment er nemt og problemfrit.

Cross-Origin Requests konfiguration

Da Azure CORS filter ikke er supported af SignaleR[94] det er derfor vigtigt at Azure CORS er slået fra, dvs ingen policies har overhoved.

Derfor er CORS politiken opsat i selve applikationen nærmere bestemt *setup.cs* med en ganske lempelig politik. Denne politik vil acceptere alle header, metoder og tillader Credentials da dette er nødvendigt for SignaleR som anvender cookie-based sticky sessions. Den vil dog kun acceptere requests som oprinder fra Frontend URL'en som den har via miljøvariable læs mere under Miljøvariable og konfiguration.

Databaseopsætning

Databasen er opsat således, at den kun accepterer forbindelser fra en udvikler maskine og backenden selv. Det er således vigtigt at IP som optræder og som har adgang gennem firewallen er opsat korrekt.

[94] https://docs.microsoft.com/en-us/aspnet/core/signalr/security?view=aspnetcore-3.1

Oprettelsen af forbindelsen mellem backend serveren og Databasen sker meget simpelt bare ved at angive connection-strengen til databasen som en miljøvariabel. Connection-strengen indeholder alle oplysninger, som er nødvendige for succesfuldt at oprette forbindelsen.

FeedbackDB databasen er på nuværende tidspunkt til at have en kapacitet på 2GB, dette er selvfølgelig gjort for at holde omkostningerne for kunden nede og kan nemt skaleres op va Azures platform.

Databaseopsætning i lokalt miljø

Såfremt projektet køres på en windows computer kan localDB anvendes uden videre. På Mac platformen kan projektet køres ved at anvende Docker til at køre et billede som indeholder MSSQL serveren[95].

Database migrations

I tilfælde af at det underliggende database skema skal modificeres kan en migration foretages.

Efter en ændring af skemaet kan der automatisk genereres en migration ved at køre følgende kommando

```
dotnet ef migrations add NAME_OF_MIGRATION -v
```

For at lade migration tage effekt på databasen skal der dernæst køres

```
dotnet ef database update
```

Database migrationerne lægger lige nu pladsceret i Startup projektet WebApi i mappen migrations, det ville nok give bedre mening til at ligge under data for fremtiden.

95 https://docs.microsoft.com/en-us/sql/linux/quickstart-install-connect-docker?view=sql-server-ver15&pivots=cs1-bash

Database seeding

Under udviklingen af systemet er databasen blevet nulstillet flere gange for at gøre denne process nemmere og hurtigere er der lavet et script som sørger for at seede[96] databasen med essentiel data.

Dette gør også testing processen væsentlig nemmere, da der er det samme datagrundlag for at teste ved hver iteration af databasen.

Det betyder også at fx. unit testing processen kan arbejde med samme data i hver enkelt adskilt test.

Koden til seeding af databasen er lokaliseret i data projektet under seedings. Her findes klassen *DBSeeding.cs* som indeholder selve seeding koden.

Koden seeder følgende data hvor en del af dataene er nødvendig for systemet kan fungere korrekt.

- Virksomheder
    - Spinoff, Virksomheds ID 1.
    - Test Aps, som navnet antyder blot en anden virksomhed for at kunne teste systemet med multiple virksomheder.
- Kategorier
    - Undervisning
    - Foredrag
    - Møder
- Spørgsmåls sæt
    - Undervisning
    - Foredrag
    - Møder
- Roller
    - Admin
    - Vadmin
    - Facilitator
- Brugere
    - Admin bruger, Spinoff
    - Virksomhedsadministrator, Spinoff
    - Facilitator, Spinoff
    - Facilitator og Virksomhedsadministrator, Test Aps

De data som er nødvendige for at systemet kan fungere korrekt er Spinoff virksomheden med virksomhed ID 1, samt rollerne. Derudover vil det være meget upraktisk ikke at have mindst en Admin bruger.

[96] https://docs.microsoft.com/en-us/ef/core/modeling/data-seeding

Seedningen af disse data finder kun sted såfremt de ikke allerede er i databasen, det vil sige, at hvis databasen udskiftes vil disse data blive lagt i databasen ved første opstart af projektet. Seeding metoden i *DBSeeding.cs* bliver kørt i *Program.cs,* altså ved opstarten løsningen.

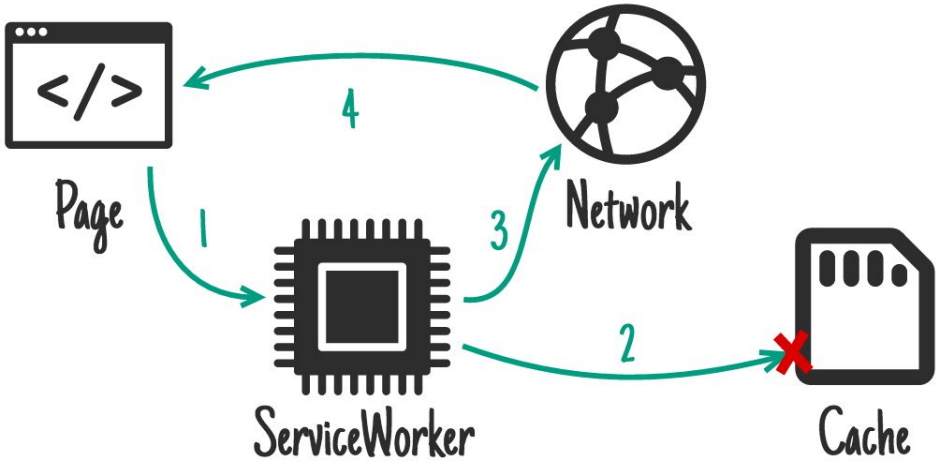
Hele seeding metoden anvender blot sub procedure til lave den fulde seeding så i tilfælde af at nogle af dataene ikke skal seedes i databasen eller flere skal tilføjes kan det nemt gøres ved at tilføje en ny procedure.

Specielt virksomhed med id 1 er Spinoff og alle kan se Spinoff's spørgsmålsset.

Bilag 22 - Caching strategier

Cache First

Cache-first er, som navnet antyder, at når service workeren bliver bedt om en ressourse, vil den altid spørge cachen først. Såfremt der findes en cached udgave, vil denne blive brugt, og der vil slet ikke blive sendt nogle forespørgsel over netværket[97].



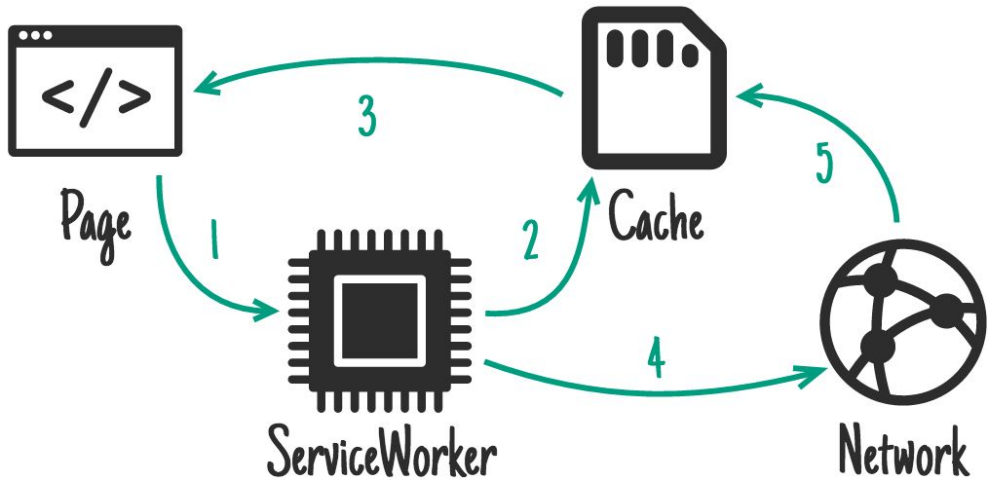
Cache First (Illustration fra workbox dokumentation)

Denne strategi er at foretrække for ressourcer, hvor det ikke er kritisk at have en ældre version, og det kan overlades til Service Workeren at opdatere ressourcer hen ad vejen. Opdateringen vil ske efter et tidsinterval er overskredet, angivet i konfigurationen eller resursen er blevet forespurgt et bestemt antal gange. Det kan dog tilføjes at denne caching strategi kun er anvendt for skrifttyper hentet fra eksterne sider og dette hvilket denne PWA ikke i skrivende stund anvender.

Stale while revalidate

Stale while revalidate strategien prøver at give brugeren svar på en forespørgsel så hurtigt som muligt ved som det første at spørge cachen efter en udgave af resursen og dernæst opdatere resursen ved et netværkskald. Såfremt resursen ikke er i cache må den selvfølgelig vente på netværkskaldet.

[97] https://developers.google.com/web/tools/workbox/modules/workbox-strategies



Cache First (Illustration fra workbox dokumentation)

Denne strategi vil derfor være egnet til ressourcer, som i højere grad end cache first har brug for den nyeste version, men hvor det stadig ikke er kritisk, at det er den nyeste som først anvendes. Dette vil fx sige, at hvis et billede bliver udskiftet, vil det gamle billede blive brugt, lige indtil netværkskaldet har returneret den nye udgave.

Default cache configuration anvender denne strategi på javascript filer, stylesheets, statiske skrifttyper (findes som fil i kodebasen) og data filer så som csv og json.

Network first

Network First prøver at fetche resursen og ved fejl falde tilbage på cachen, det er denne strategi som bliver brugt til alle kald til REST API'en. Der er dog sat et meget lagt fallback interval således at det kun er hvis serveren fejler, eller responstiden er over 10 sek, at cachen reelt vil blive brugt.

Af andre strategier kan nævnes Network Only, der kan bruges i tilfælde, hvor det kun er acceptabelt at kunne hente den nyeste udgave af resursen direkte fra den oprindelige server. Cache Only, som kun beror på cachen, som typisk vil være opnået ved precaching.

Bilag 23 - Progressive webapplikationer som teknologi

En PWA er en app baseret på webteknologier såsom HTML 5, css og javascript, og som anvender en service worker, der vil blive beskrevet senere, samt en app-distributionsmetode, som kunne være bundlet i en form for native skald (webview) eller som Web manifest som en browser kan fortolke, hvilket er den metode som anvendes i dette projekt.

Siden en PWA er udviklet til at køre i en browser/web view vil den være multiplatform af natur, da der findes en browser på langt de fleste enheder i dag, og en stor del af dem i et eller andet omfang supporter PWA'er.

PWA teknologien er mere er et design mønster end en enlig specifikation, og bygger på standard web teknologier[98]. Dette betyder også at forskellige browsere implementerer de nødvendige features til PWA en smule forskelligt. Disse forskelle og svingende support for PWA kan dog være en udfordring. Eksempelvis er der forskel på, hvor meget de forskellige platforme lader app'en cache. Browseren Safari til IOS supporterer kun 50MB data[99]. Et andet eksempel på endnu ikke standardiserede features kan findes i afsnittet Installerbar.

PWA forsøger at kombinere features traditionelt forbundet med udvikling med native SDK'er til en given platform med web udviklings standarder.

Ved native udvikling vil man typisk distribuere sin applikation via en app store. Denne tilgang kan både være en længere proces, da koden i nogle tilfælde skal kvalitetssikres/trussel undersøges, før den kan findes i app storen, og det kan være dyrt for firmaet bag, da der samtidig skal betales for at være registreret som udvikler og gives en del af omsætningen i appen tilbage til distributøren.

Dette problem har PWA'er slet ikke, da de meget simpelt kan lægges op på en server som en hjemmeside og distribueres den vej igennem. Yderligere har PWA en klar fordel at appen kan bruges i browseren uden installation, hvilket især kan betragtes som en fordel i brugerscenarie U1, hvor brugeren giver feedback. På den måde bliver brugeren ikke tvunget til at hente og installere en app for at give feedback én gang.

Ligeledes er opdateringen af app'en også meget nem, da dette blot er at udskifte filerne som på serveren med den nye version. Brugerne vil derefter gradvist få den nye udgave, mens de bruger det. Hvordan dette sker er beskrevet under Caching afsnittet.

[98] https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
[99] https://love2dev.com/blog/what-is-the-service-worker-cache-storage-limit/

For nogle forretninger og ikke mindst brugere kan det være en deal breaker ikke at kunne have sin app i app storen, og i det tilfælde er PWA'er ikke en teknologi man bør vælge.

Google har set dette problem og gjort det muligt at få PWA på google play store via Trusted Web Activities, hvor man bundler sin PWA i et web view/web aktivitet[100].

Apple tillader ikke PWA'er i deres app store og giver ikke mulighed for at bundle så det kan overholde deres retningslinjer[101].

I og med at PWA kan siges og være sand cross-platform og der dermed kun skal udvikles en udgave af appen kan dette betyde væsentlig reduceret pris for udviklingen af en PWA contra både native og hybride løsninger. Det kan dog stadig være svært at få app'en til at fremstå med en høj kvalitet og brugeroplevelse på alle platforme grundet, hvor alsidig den er, hvor en af faktorerne også er beskrevet under afsnittet Styling.

App'en kan nemt blive fundet, da den vil blive indekseret af google og andre søgemaskiner og har gode egenskaber for Search Engine Optimization (SEO) især med muligheden for at SSR dele af applikationen.

[100] https://developers.google.com/web/android/trusted-web-activity
[101] https://developer.apple.com/news/?id=01212020a

Litteraturliste

"185448 – getUserMedia Not Working in Apps Added to Home Screen That Run in Standalone Mode." n.d. Accessed June 24, 2020. https://bugs.webkit.org/show_bug.cgi?id=185448.

"Advanced Features: Dynamic Import | Next.js." n.d. Accessed June 24, 2020. https://nextjs.org/docs/advanced-features/dynamic-import.

ajcvickers. n.d. "Testing Code That Uses EF Core - EF Core." Accessed June 24, 2020. https://docs.microsoft.com/en-us/ef/core/miscellaneous/testing/.

AndriySvyryd. n.d. "Relationships - EF Core." Accessed June 24, 2020. https://docs.microsoft.com/en-us/ef/core/modeling/relationships.

Apple Inc. n.d. "Tab Bars - Bars - iOS - Human Interface Guidelines - Apple Developer." Accessed June 24, 2020. https://developer.apple.com/design/human-interface-guidelines/ios/bars/tab-bars/.

Archiveddocs. n.d. "Use the Outlook REST API (version 2.0)." Accessed June 24, 2020. https://docs.microsoft.com/en-us/previous-versions/office/office-365-api/api/version-2.0/use-outlook-rest-api.

ardalis. n.d. "Overview of ASP.NET Core MVC." Accessed June 24, 2020. https://docs.microsoft.com/en-us/aspnet/core/mvc/overview.

"AutoMapper." n.d. Accessed June 24, 2020. https://automapper.org/.

Bähr, Katharina. 2017. "How TypeScript Helps Us to Develop Great Software - Zühlke Blog." Zühlke Blog. October 2, 2017. https://www.zuehlke.com/blog/why-should-consider-typescript-for-your-project/.

"BeforeInstallPromptEvent." n.d. MDN Web Docs. Accessed June 24, 2020. https://developer.mozilla.org/en-US/docs/Web/API/BeforeInstallPromptEvent.

Bhagat, Hitesh Raj, and Karan Bajaj. 2018. "The 18:9 Display Dilemma: Will the New Smartphone Screens Make Our Lives Easier or Do the Opposite?" The Economic Times. Economic Times. January 26, 2018. https://economictimes.indiatimes.com/magazines/panache/the-189-display-dilemma-will-the-new-smartphone-screens-make-our-lives-easier-or-do-the-opposite/articleshow/62662023.cms.

Bibile, Udara. 2019. "Unit of Work for ASP.NET Core - Udara Bibile - Medium," December. https://medium.com/@chathuranga94/unit-of-work-for-asp-net-core-706e71abc9d1.

"Bottom Navigation - Material Components for Android." n.d. Material Components for Android. Accessed June 24, 2020. https://material.io/develop/android/components/bottom-navigation/.

Bradley, John, Nat Sakimura, and Michael B. Jones. 2015. "JSON Web Token (JWT)," May. https://tools.ietf.org/html/rfc7519.

bradygaster. n.d. "ASP.NET Core SignalR Supported Platforms." Accessed June 24, 2020a. https://docs.microsoft.com/en-us/aspnet/core/signalr/supported-platforms.

———. n.d. "Authentication and Authorization in ASP.NET Core SignalR." Accessed June 24, 2020b. https://docs.microsoft.com/en-us/aspnet/core/signalr/authn-and-authz.

———. n.d. "Introduction to SignalR." Accessed June 24, 2020c. https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr.

———. n.d. "Manage Users and Groups in SignalR." Accessed June 24, 2020d. https://docs.microsoft.com/en-us/aspnet/core/signalr/groups.

Brzóska, Bazyli. n.d. "Hashids." Hashids. Accessed June 24, 2020. https://hashids.org/.

cephalin. n.d. "Overview - Azure App Service." Accessed June 24, 2020.

https://docs.microsoft.com/en-us/azure/app-service/overview.
Chandu. 2019. “Are Single Page Applications Still Relevant in 2020?,” August. https://medium.com/commutatus/seo-in-single-page-applications-csr-vs-ssr-e342d7cc69b.
“Chapter 10: MoSCoW Prioritisation.” n.d. Accessed June 24, 2020. https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation.
“Chart.js | Open Source HTML5 Charts for Your Website.” n.d. Accessed June 24, 2020. https://www.chartjs.org/.
Copes, Flavio. 2019. “The Next.js App Bundles.” Flaviocopes.com. November 27, 2019. https://flaviocopes.com/nextjs-app-bundles/.
“Cross Platform React Mobile UI Controls | Mobiscroll.” n.d. Accessed June 24, 2020. https://mobiscroll.com/react.
“Databeskyttelsesforordningen.” n.d. Accessed June 24, 2020. https://eur-lex.europa.eu/legal-content/DA/TXT/PDF/?uri=CELEX:32016R0679&from=DA.
“Designing Websites for iPhone X.” 2017. WebKit. September 22, 2017. https://webkit.org/blog/7929/designing-websites-for-iphone-x/.
“Destructuring Assignment.” n.d. MDN Web Docs. Accessed June 24, 2020. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment.
dotnet-bot. n.d. “ClaimTypes Class (System.Security.Claims).” Accessed June 24, 2020a. https://docs.microsoft.com/en-us/dotnet/api/system.security.claims.claimtypes.
———. n.d. “IdentityUser Class (Microsoft.AspNetCore.Identity.EntityFrameworkCore).” Accessed June 24, 2020b. https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.identity.entityframeworkcore.identityuser.
———. n.d. “RequiredAttribute Class (System.ComponentModel.DataAnnotations).” Accessed June 24, 2020c. https://docs.microsoft.com/en-us/dotnet/api/system.componentmodel.dataannotations.requiredattribute.
———. n.d. “System.ComponentModel.DataAnnotations Namespace.” Accessed June 24, 2020d. https://docs.microsoft.com/en-us/dotnet/api/system.componentmodel.dataannotations.
———. n.d. “UserManager Class (Microsoft.AspNetCore.Identity).” Accessed June 24, 2020e. https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.identity.usermanager-1.
“Email Delivery Service.” n.d. SendGrid. Accessed June 24, 2020. https://sendgrid.com/.
“Env().” n.d. MDN Web Docs. Accessed June 24, 2020. https://developer.mozilla.org/en-US/docs/Web/CSS/env.
Felix, Jose. 2020. “Why Use Bottom Navigation in Mobile Websites.” DEV Community. DEV Community. February 27, 2020. https://dev.to/jfelx/why-bottom-navigation-in-mobile-pages-2j9l.
“Fetch API.” n.d. MDN Web Docs. Accessed June 24, 2020. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API.
fingerprintjs. n.d. “fingerprintjs/fingerprintjs2.” GitHub. Accessed June 24, 2020. https://github.com/fingerprintjs/fingerprintjs2.
Firtman, Maximiliano. 2020. “Progressive Web Apps in 2020 - Maximiliano Firtman - Medium,” January. https://medium.com/@firt/progressive-web-apps-in-2020-c15018c9931c.
“Get Started | Workbox | Google Developers.” n.d. Google Developers. Accessed June 24,

2020. https://developers.google.com/web/tools/workbox/guides/get-started.
“Getting Started with Media Devices | WebRTC.” n.d. WebRTC. Accessed June 24, 2020. https://webrtc.org/getting-started/media-devices.
“Higher-Order Components – React.” n.d. Accessed June 24, 2020. https://reactjs.org/docs/higher-order-components.html.
“Hooks API Reference – React.” n.d. Accessed June 24, 2020. https://reactjs.org/docs/hooks-reference.html.
inconshreveable. n.d. “Ngrok - Secure Introspectable Tunnels to Localhost.” Accessed June 24, 2020. https://ngrok.com/.
“Introducing Hooks – React.” n.d. Accessed June 24, 2020. https://reactjs.org/docs/hooks-intro.html.
“Introduction · MobX.” n.d. Accessed June 24, 2020. https://mobx.js.org//index.html.
Ionic. n.d. “UI Components - Ionic Documentation.” Ionic Docs. Accessed June 24, 2020. https://ionicframework.com/docs/components.
“JavaScript End to End Testing Framework.” n.d. JavaScript End to End Testing Framework | Cypress.io. Accessed June 24, 2020. https://www.cypress.io/.
“Json.NET - Newtonsoft.” n.d. Accessed June 24, 2020. https://www.newtonsoft.com/json.
Kukic, Adnan, and By Core. n.d. “Cookies vs. Tokens: The Definitive Guide - DZone Integration.” Dzone.com. Accessed June 24, 2020. https://dzone.com/articles/cookies-vs-tokens-the-definitive-guide.
Larman, Craig. 2002. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*. Prentice Hall Professional.
“Learn | Next.js.” n.d. Accessed June 24, 2020a. https://nextjs.org/learn.
———. n.d. Accessed June 24, 2020b. https://nextjs.org/learn.
“localForage.” n.d. Accessed June 24, 2020. https://localforage.github.io/localForage/.
“Making PWAs Work Offline with Service Workers.” n.d. MDN Web Docs. Accessed June 24, 2020. https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Offline_Service_workers.
“Material Design.” n.d. Material Design. Accessed June 24, 2020. https://material.io/components/navigation-drawer#exposed-dropdown-menu.
MicrosoftDocs. n.d. “Azure App Service Linux Container - Web Socket Handshake Error 503 · Issue #49245 · MicrosoftDocs/azure-Docs.” GitHub. Accessed June 24, 2020. https://github.com/MicrosoftDocs/azure-docs/issues/49245.
“Mobile Vs. Desktop Internet Usage (Latest 2020 Data).” n.d. BroadbandSearch.net. Accessed June 24, 2020. https://cdn.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics.
“Modifiers · MobX.” n.d. Accessed June 24, 2020. https://mobx.js.org//index.html.
mvllow. n.d. “Mvllow/next-Pwa-Template.” GitHub. Accessed June 24, 2020. https://github.com/mvllow/next-pwa-template.
nishanil. n.d. “Designing the Infrastructure Persistence Layer.” Accessed June 24, 2020. https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design.
“Npm: @aspnet/signalr.” n.d. Npm. Accessed June 24, 2020. https://www.npmjs.com/package/@aspnet/signalr.
“Npm: Next-Pwa.” n.d. Npm. Accessed June 24, 2020. https://www.npmjs.com/package/next-pwa.
“Npm: React-Chartjs-2.” n.d. Npm. Accessed June 24, 2020. https://www.npmjs.com/package/react-chartjs-2.
“Npm: React-Export-Excel.” n.d. Npm. Accessed June 24, 2020. https://www.npmjs.com/package/react-export-excel.

“Npm: React-Lottie.” n.d. Npm. Accessed June 24, 2020. https://www.npmjs.com/package/react-lottie.
“Npm: React-Qr-Reader.” n.d. Npm. Accessed June 24, 2020. https://www.npmjs.com/package/react-qr-reader.
“Ping Identity.” n.d. Accessed June 24, 2020. https://www.pingidentity.com.
“Precache Files | Workbox.” n.d. Google Developers. Accessed June 24, 2020. https://developers.google.com/web/tools/workbox/guides/precache-files.
“Pro vs Free Version.” n.d. Accessed June 24, 2020. https://docs.fingerprintjs.com/pro/pro-vs-free.
Rick-Anderson. n.d. “Introduction to Identity on ASP.NET Core.” Accessed June 24, 2020. https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity.
“Routing: Shallow Routing | Next.js.” n.d. Accessed June 24, 2020. https://nextjs.org/docs/routing/shallow-routing.
“Selectors Level 4.” n.d. Accessed June 24, 2020. https://www.w3.org/TR/selectors/#root-pseudo.
“Service Worker API.” n.d. MDN Web Docs. Accessed June 24, 2020. https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API.
Sethi, Ankur. 2018. “The Baseline Costs of JavaScript Frameworks - Ankur Sethi - Medium,” November. https://medium.com/@ankurs3thi/the-baseline-costs-of-javascript-frameworks-f768e2865d4a.
shadowwalker. n.d. “Next-Pwa Cache.” GitHub. Accessed June 24, 2020. https://github.com/shadowwalker/next-pwa.
“Spinoff.” n.d. Spinoff. Accessed June 24, 2020. https://www.spinoff.nu.
“Top Front-End Frameworks in 2020 | Existek Blog.” 2020. January 14, 2020. https://existek.com/blog/top-front-end-frameworks-2020/.
“Uptime Robot | Free Website Monitoring.” n.d. Uptime Robot. Accessed June 24, 2020. https://uptimerobot.com/.
“Using CSS Custom Properties (variables).” n.d. MDN Web Docs. Accessed June 24, 2020. https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties.
“Using the State Hook – React.” n.d. Accessed June 24, 2020. https://reactjs.org/docs/hooks-state.html.
“Virtual Environments for GitHub-Hosted Runners - GitHub Help.” n.d. Accessed June 24, 2020. https://help.github.com/en/actions/reference/virtual-environments-for-github-hosted-runners.
“Window: Appinstalled Event.” n.d. MDN Web Docs. Accessed June 24, 2020. https://developer.mozilla.org/en-US/docs/Web/API/Window/appinstalled_event.
zuckerthoben. n.d. “Get Started with Swashbuckle and ASP.NET Core.” Accessed June 24, 2020. https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle.