



Support Web Desk

Datamatiker hovedopgave

Andreas Bonke & Nikolaj Bang Nielsen

Erhvervsakademi Sjælland

Vejleder: Jamshid Eftekhari

periode: 31/10/2017-05/01/2018

Indholdsfortegnelse

	0
Indholdsfortegnelse	1
Indledning	5
Problemformulering	6
Methodology	7
Scrum	7
Extreme programming	7
Opgavens struktur	9
Sprint 0	10
Vision statement	10
Risikoanalyse	10
Technical spike	11
Arkitektur	12
Systemkrav	14
Andre krav	14
Setup Environment	15
Teamets roller	16
Product Backlog	17
User stories	17
Planning poker	21
Prototype	22
Sprint 0 refleksion	24
Sprint 1	26
Sprint planning	26
Løbende testing	26
Sprint backlog	27
Scrum møder	28
Opsummering af scrum møder	28
Sprint review	29
Burndown chart	30
Sprint Retrospective	31
Tekniske detaljer	31
Code First Design database til tickets	32
Migrations	33

Oprette API endpoint til at trække data ud af database	34
Design angular komponent til visning af tickets	35
Angular service som kan benytte sig af API endpoint	35
Arbejdet for at opnå acceptance kriterierne	35
Testing	36
Sprint 2	38
Sprint planning	38
Sprint Backlog	38
Scrum møder	39
Opsummering af Scrum møder	39
Sprint Review	40
Brugertest	40
Burndown Chart	42
Sprint Retrospective	42
Tekniske detaljer	42
Oprette brugersystem	43
Design bruger authorization til controllere	43
Design angular komponent til login	44
Angular komponent for bruger forside	44
Oprette et API endpoint til at trække tickets ud til bestemt bruger	45
Testing	45
Sprint 3	46
Sprint planning	46
Sprint Backlog	47
Scrum møder	48
Opsummering af Scrum møder	48
Tekniske detaljer	48
Baggrundsjobs	49
Mails gemmes og processeres	50
Mail service til afsendelse af mails og formalia	50
Testing	51
Sprint 4	52
Sprint planning	52
Sprint Backlog	52
Opsummering af scrum møder	53
Sprint Review	54

Brugertest	54
Burndown Chart	55
Sprint Retrospective	56
Tekniske detaljer	57
Beskeder og noter implementeringen	57
Testing	58
Sprint 5	59
Sprint planning	59
Sprint Backlog	59
Opsummering af scrum møder	60
Sprint review	61
Burndown chart	61
Sprint retrospective	62
Tekniske detaljer	63
Implementer search component med søgefunktionalitet	63
Implementering af backend søge funktioner	64
Testing	65
Sprint 6	66
Sprint planning	66
Sprint Backlog	67
Opsummering af scrum møder	67
Sprint review	68
Brugertest	68
Burndown chart	69
Sprint retrospective	70
Tekniske detaljer	70
Status skift i view componentet	71
Status skift endpoints	72
Testing	72
Konklusion	75
Refleksion	76
Litteraturliste	79
Bilag	80
Prototypes	80
Bilag 1.1	80
Bilag 1.2	81

Bilag 1.3	82
Bilag 2	83
Shared Vision	83
Bilag 3	86
Bilag 4	89
Bilag 5	90

Indledning

I løbet af vores praktikforløb stødte vi begge på problemstillinger, der kunne løses og være et godt emne, for vores afsluttende hovedopgave. Efter at have diskuteret de forskellige muligheder, blev vi enige om at det var ideen om en web applikation, der kunne fungere som et værktøj i hverdagen, til at holde styr på support henvendelser, der var den mest spændende. Ideen til denne opgave opstod i forbindelse med Andreas' praktikforløb, da han i Synergi reklame- og webbureau, diskuterede problemstillingen om manglende overblik med en anden programmør.

Synergi er et reklame- og webbureau, som blev etableret d. 1 januar 2003 af direktør, Thomas Hansen. Det var på daværende tidspunkt et lille reklamebureau, med Thomas og en enkelt grafisk designer. D. 1/1-2010 begyndte Synergi at udvikle hjemmesider. Med salget af hjemmesideløsninger kom også andre ydelser, som for eksempel support til de solgte hjemmesider. I starten gik det fint med support, da kunderne blot henvendte sig via mail når der var problemer. Men efterhånden som Synergi voksede gjorde deres kundekartotek det også. Pludseligt var det besværligt og tidskrævende, at organisere support henvendelser i en mail indbakke. Det er her at vi, som udviklere, kommer ind i billedet og kan være med til at skabe en løsning, der kan mitigere store problemer i fremtiden.

For at se det endelige resultat, er her linket til github:

<https://github.com/Bang0123/Project-Support-Web-Desk>

Problemformulering

Ønsket om et professionelt overblik og tid til mere er noget, som de fleste kan genkende, når det kommer til deres arbejdsliv. Hos Synergi reklame- og webbureau har man erkendt at begge dele er manglende, når det kommer til deres support sager. Når deres kunder henvender sig for at få support, mangler Synergi overblik over sagerne internt.

På nuværende tidspunkt har de alle deres support henvendelser liggende i en mail indbakke. Det gør at der ikke er overblik og i sidste ende koster det tid, at skulle finde hoved og hale i det hele.

Vi vil derfor skabe et produkt, som gør det nemt og overskueligt at håndtere support sager, så der bliver mere tid til andre opgaver, som i sidste ende er det, der skaber omsætning i virksomheden. Produktet vil munde ud i en webapplikation med nogle sider der viser support sager, på en overskuelig måde, der gør det let at gå til de forskellige opgaver. Dette vil vi gøre så vidt det er muligt, i samarbejde med Bo fra Synergi, så vi kommer så tæt på hans vision for produktet som muligt.

Til rapporten har vi opstillet nogle problemstillinger, som vi vil forsøge at besvare ved hjælp af værktøjer og de evner, som vi har tilegnet os gennem undervisningen.

Vi vil blandt andet benytte os af analyser, modeller og udviklingsmetoder. Til sidst vil vi i rapporten reflektere over forløbet, og konkludere på hvordan vi har løst følgende problemstillinger:

1: "Hvordan kan man ved hjælp af en web applikation få overblik over support efterspørgsler i en moderne virksomhed?"

2: "Hvordan kan man ved hjælp af softwareudviklings teori sikre at man kommer frem til et godt resultat?"

Methodology

I løbet af uddannelsen, er vi blevet præsenteret for flere måder at udvikle og styre vores projekter på. Heriblandt var der Unified Process, Scrum og Extreme programming.

I tidligere projekter har vi prøvet at arbejde med tidligere nævnte methodologies og efter at have diskuteret mulighederne er vi blevet enige om at benytte os af Scrum og Extreme programming. Herunder vil vi kort redegøre for vores valg.

Scrum

I forbindelse med tidligere projekter, har vi erfaret at scrum er et fantastisk "framework" til at styre og have overblik over et projekt. Derudover komplementerer det extreme programming rigtig godt. Vi har valgt scrum, da det var et ønske for os at have god kontakt med product owner, og inkludere ham så meget som muligt i udviklingen af produktet.

Scrum er godt at arbejde med i agil udvikling og gør det også muligt at komme med nye krav specifikationer løbende. Dette regner vi også med kommer os til gode i løbet af vores sprints, da vi undervejs kunne komme på nye ideer, der ville øge kvaliteten af produktet.

En anden god ting ved scrum er at, vi selv kan bestemme hvor lange sprints vi ønsker. Dette kommer til at fungere virkelig godt med extreme programming, da vi ønsker at komme med små releases hver uge, så product owner også kan se at der sker fremskridt.

Extreme programming

Extreme programming arbejder ud fra en masse utroligt brugbare principper. Men der hvor extreme programming er mangelfuldt, er på projektstyrings plan. Som nævnt ovenfor imødekommer vi de udfordringer, der kunne være ved extreme programming, som enkeltstående methodology, ved at komplimentere det med Scrum. På den måde kan vi benytte os af alle fordelene ved extreme programming, uden at bekymre os om vi nu overholder releases og andre tekniske ting, som vi så har styr på igennem scrum.

Derudover er der specielt to af de values, som extreme programming bygger på som vi sætter pris på i denne sammenhæng, og det er kommunikation og feedback. Det er vigtigt for os at have god kommunikation med product owner, da produktet stadig er en ny ide for ham også. Igen er det to gode punkter, som går godt i spænd med scrum projekt styring.

En anden ting som extreme programming gør utroligt godt er test driven development (TDD). Dette er en disciplin som vi føler kræver en masse erfaring at mestre og vi har derfor valgt ikke at opstille test cases før hver user story. Vi vil derimod i de forskellige afsnit for tekniske detaljer beskrive hvordan vi har testet koden i de komponenter som vi føler giver mening at teste.

Dette kan virke meget uortodokst men vi ser de forskellige methodologies som en slags frameworks hvor man til en hvis grad kan håndplukke de elementer som vil gøre at man i sidste ende får et godt produkt.

Opgavens struktur

De næste mange sider vil omhandle de forskellige sprints, som gruppen har været igennem. Selve opsætningen af stykkerne kan ved første øjekast virke meget slaviske, men det er der en grund til. Dette er valgt for at give læseren et indblik i, hvilken rækkefølge vi har udført de forskellige discipliner i og hvordan arbejdet skred frem.

Ser man bort fra strukturen i sprint 0, vil der i hver sprint være en overskrift med hvilken sprint der er blevet udarbejdet, og hvilke scrum events/artefakter der er blevet gjort brug af. Til sidst vil der være en lettere teknisk gennemgang af den kode, som er blevet udviklet så læseren også kan følge med i hvordan selve produktet skrider frem.

Sprint 0

Vision statement

Som det første i projekt etableringen, udarbejdede vi en shared vision. Dette gjorde vi for at være sikre på at alle involverede, vidste hvad behovet var og hvordan product owner ønskede det løst. Ud fra denne shared vision, kan vi drage nogle konklusioner om hvordan situationen er nu og hvordan situationen er ønsket i fremtiden.

Efter at have udarbejdet et vision statement, har vi fundet ud af at problemet, som tidligere beskrevet, er uoverskuelig kundesupport sager. Det påvirker medarbejderne i Synergi men også kunderne da de potentielt kan komme til at vente længere før de får den hjælp de behøver. En god løsning på problemet ville ifølge product owner være et program, eksempelvis en web applikation, som kan give et overblik over de kunder der har henvendt sig.

Løsningen skal altså være med til at bidrage til supporterens arbejdsgang, så han kan få afviklet efterspørgslen på support mere effektivt.

Derudover fik vi product owner til at komme med nogle funktioner, som skal være i web applikationen. Udover at det får product owner til at tænke over hvordan løsningen skal se ud, kan det også være med til at give ham inspiration til user stories senere i processen.

Den udarbejdede vision statement er vedhæftet i bilag og indeholder alt det ovennævnte plus andre krav til produktet.

Risikoanalyse

Efter vores shared vision udarbejdede vi en Risikoanalyse, for at imødekomme de fejl og udfordringer vi kunne komme ud for i løbet af udarbejdelsen af produktet.

Ved tabellen skal kolonnerne forstås således:

Magnitude: Defineres med en værdi fra 1 og 10. 1 er den laveste og 10 er den højeste.

Værdien er angivet ud fra chancen for at problemet opstår.

Impact: Er en indikation for hvor stor indflydelse at scenariet har på projektet. Kolonnen er angivet ud fra værdierne Lav, Mellem, Høj og Kritisk.

Indicator: Indicator er en simpel beskrivelse af indikationen for at man er ved at løbe ind i problemet.

Mitigation: Måden hvorpå vi vil undgå problemet eller bearbejde det, i tilfælde af at problemet skulle opstå.

Types: Planlægning, Teknisk, Ressourcer og Økonomi er de typer af kategorier vi deler de forskellige cases op i.

Type	Beskrivelse	Magnitude	Impact	Indicator	Mitigation
Technical	Manglende erfaring med Angular 4.	5	Kritisk.	Front end udvikling tager for lang tid.	Research om brugen af Angular i god tid.
Technical	Manglende erfaring med smtp og imap i .NET core.	8	Kritisk.	At benytte .NET core smtp og imap giver problemer.	Research, eller migrering til web kun.
Economical	Product owner stopper samarbejdet.	10	Meget Kritisk.	Synergi dropper projektet og ideen om dette system.	Projektet skal køres videre i resten af opgave forløbet.
Ressources	Tab af kildekode.	7	Kritisk.	Versionsstyring skaber fejl.	migrere til tidligere version og lave backups.
Technical	Hangfire baggrundsprocesser virker ikke.	4	Mellem.	Hangfire processer bliver blokeret af .NET Core web applikationen.	Lade controllere opdatere og hente mails efter behov.

Technical spike

som man kan se i tabellen ovenfor, kom vi i vores risikoanalyse frem til at en af de store risks ved vores projekt, var at vi ikke havde nogen erfaring med smtp og imap i .NET core. Derfor valgte vi at udføre et technical spike, for at finde ud af om det kunne lade sig gøre og dermed mitiggere fremtidige problemer. På denne måde undgår vi også at bruge en masse ressourcer på at researche en problemstilling i sprinten, så vi ikke leverer nogle releases for sent og overholder tidsplanen i vores sprints.

Et spike er en research aktivitet, som kommer fra Extreme programming¹. Det er en aktivitet som har til formål at øge udviklernes viden omkring en problemstilling de står overfor. Et spike ligger normalt midt i udviklingen af et produkt og vil så finde sted i slutningen af en iteration. Vi valgte at udføre et spike før den egentlige udviklingsfase for at mitigere chancen for at det ville have en stor impact på vores igangværende iteration, da de kun er en uge lange.

Vi brugte tid på at researche problemet på google, hvor der kom en masse eksempler på hvordan man gør i .NET, bare ikke .NET Core. Vi fandt frem til et framework kaldet MimeKit² og MailKit, samt en masse eksempler på hvordan man kan manipulere med imap. Disse eksempler var dog for længst forældede, men frameworket var for nylig blevet opdateret og kunne bruges sammen med en .NET Core applikation. Med et eksempel som udgangspunkt har vi kunne opnå, at hente mails fra en indbakke og markere dem som læste. Vi har også set på Hangfire³, da vores ide er at mails skal hentes af systemet i baggrunden, hvor den lagrer data i en MS SQL database. Hangfire har også en version kompatibel med .NET Core, dog er dokumentationen ikke helt opdateret på nogle områder, men det har været nogenlunde nemt at komme i gang med. Hangfire køres i sin helt egen process ved siden af Web applikationen, så denne løsning vil ikke påvirke brugerens oplevelse.

Arkitektur

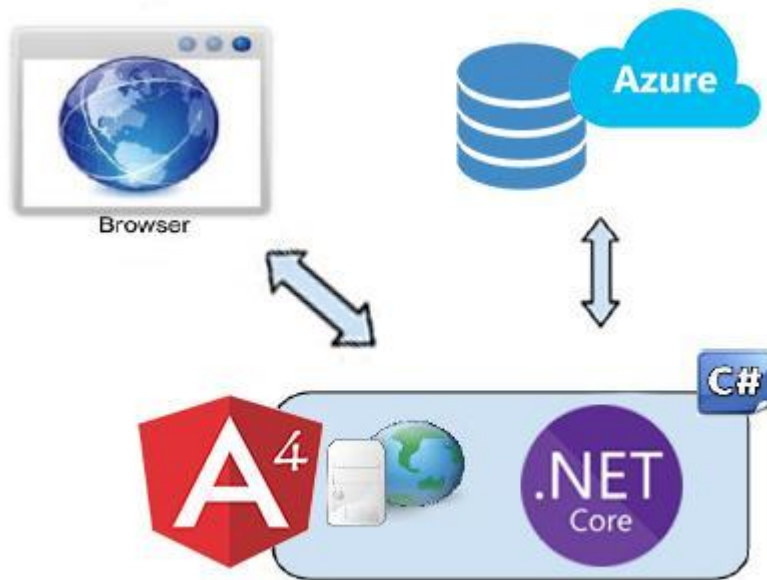
udover vores tekniske spike, havde vi også andre tekniske overvejelser. Heriblandt var det arkitekturen for applikationen. I dette projekt har vi udtænkt en arkitektur efter mange overvejelser. Denne arkitektur består i bund og grund af en arkitektur som indeholder:

- MSSQL Database hos Azure Cloud Service.
- ASP.NET Core 2.0 Backend.
- Frontend i Angular 4/5.
- Færdig Web applikationen hostet af fx Azure Cloud Service.

¹ <http://www.scaledagileframework.com/spikes/>

² <http://www.mimekit.net/> fully featured Mail framework for .NET og .NET Core

³ <https://www.hangfire.io/>



I vores overvejelser har vi valgt at bruge disse værktøjer, med udgangspunkt i en Web MVC Applikation. Vi har valgt .NET Core, da det efterhånden er ved at være godt med i version 2.0⁴. Det mangler stadig en del biblioteker fra det fulde .NET framework, men det yder god performance og er cross-platform. Vi havde overvejelser om at benytte Laravel 5.5⁵, da Laravel har været ude i længere tid og har rigtig mange kode biblioteker til fri afbenyttelse. Dog har kun Nikolaj erfaring med dette PHP framework, så valget faldt ikke på denne teknologi. Derudover overvejede vi også Golang⁶, med web framework GoBuffalo⁷, dog er dette framework så nyt, at vi ikke ville være helt sikre i hvad vi begav os ud i. GoBuffalo ville dog have være ekstremt hurtigt, og så har sproget en native webserver og kan compiles til cross-platform.

Til Database blev vi hurtige enige om at benytte Azures database⁸, da vi ellers selv skulle finde et alternativt dyrere host til fx en postgres eller mysql database. Samt har vi mulighed for at have den hostet af deres cloud services, så vi ikke hver især i gruppen, har en lokal database. Dog har Azure en ret begrænset størrelse på 32 MB, så vi har oprettet en AWS⁹ MS SQL database til backup, hvis vi behøver en større database.

Til Frontend overvejede vi kort at benytte de indbyggede templating systemer, som hver framework tilbød. Dette gik vi hurtigt væk fra og tænkte at, vi ville udfordre os selv og lave en Angular App. Versionen af angular blev også diskuteret lidt, da Nikolaj kun havde erfaring

⁴ <https://blogs.msdn.microsoft.com/dotnet/2017/08/14/announcing-net-core-2-0/>

⁵ <https://laravel.com/>

⁶ <https://golang.org/>

⁷ <https://gobuffalo.io/>

⁸ <https://azure.microsoft.com/da-dk/services/sql-database/>

⁹ <https://aws.amazon.com/>

med AngularJS 1.6¹⁰. Vi endte med at vælge versionen Angular 4¹¹, da .NET Core tilbød en template der gjorde opsætningen meget nem. Men det udelukkes ikke at opgradere til Angular 5, da Angular 5 er en “major release”, så der er mange kode biblioteker der bliver skrevet til denne version, og ikke Angular 4 fremover.

Systemkrav

Med udgangspunkt i vores Shared Vision, som vi udarbejdede i samarbejde med Bo, kan udviklingsholdet komme frem til følgende krav til systemet:

- Database til lagring af data.
- Hente data fra mailbox.
- Kan sende mails.
- Offentlig tilgængelig web applikation.
- Brugersystem.
- Visning af supportsager.
- Mulighed for at svare på supportsager.
- Søgning af specifikke sager.

Andre krav

Udover de ovenstående krav til systemet er der også andre krav som gerne skulle opfyldes.

- Usability.
Brugen af systemet skal være nemt og let tilgængelig.
Brugergrænsefladen skal være let at benytte.
- Portability.
Web applikationen skal nemt kunne tilgås på flere platforme.
Den skal være Restful, så der nemt kunne laves en native App til IOS, Android, Windows eller Mac.
- Fault tolerance.
Systemet skal helst kunne køre videre med dens hjertesag, selvom der opstår fejl i andre mindre dele af systemet.
- Simple.
Systemet skal være meget simpelt, så den på sigt kan integreres med andre systemer.

¹⁰ <https://angularjs.org/>

¹¹ <https://angular.io/>

Setup Environment

I gennem projekt etableringen har vi i gruppen diskuteret forskellige teknologier, som hver især har potentiale, og udvalgt dem vi foreløbigt kan planlægge at vi benytter. Frameworks og teknologier indebærer:

- ASP.NET Core 2.0, MS SQL og Angular 4/5
- Visual studio 2017 og Visual studio Code
- Microsoft Azure
- Trello
- Google drev
- Git og github

ASP.NET Core 2.0, MS SQL og Angular 4/5

I vores arkitektur har vi valgt at benytte ASP.NET Core 2.0 framework til at udvikle vores backend i. Denne teknologi giver også nem mulighed for brug af Microsofts cloud service Azure. Dertil har vi valgt MS SQL database, da denne database type er nem at komme i gang med på azure, samt at bruge i .NET Core med entity framework core.

Til frontend har vi valgt at udfordrer os selv og lave frontend som en angular 4/5 app, ved release af .NET Core 2.0 kom der en template for nem opsætning af sådan en app, når man benytter .NET Core framework.

Ved valget af disse teknologier benytter vi C# i ASP.NET core framework, Typescript til at udvikle funktion i angular 4/5 appen og entity framework core til at manipulere med MSSQL databasen.

Visual Studio 2017 og Visual Studio Code

Vi har valgt at benytte Visual Studio 2017 IDE, denne IDE understøtter både C# og Typescript, men er ret så tung for computeren, hvis der blot skal laves små rettelser. Visual Studio 2017 IDE giver også mange værktøjer til at forbinde med database og udvikle applikationer tilpasset til Azure.

For lettere rettelser har vi besluttet og for at bruge text editoren Visual Studio Code, den er lynhurtig og der er mulighed for mange forskellige plugins, som gør at den også kan bruges i udviklingen.

Microsoft Azure

Igennem Microsoft Imagine kan vi bruge App services fra Microsoft Azure, dette kan gøre at web applikationen ligger på internettet frem for vores egne computere, når vi skal vise en demonstration af det færdige produkt. Derudover vil vores database være hostet hos Azure,

så vi i udviklingsprocessen har 1 database hos Azure, vi begge kan tilgå uanset hvor vi befinder os.

Trello

Vi har valgt at benytte trello som et Kanban Board, dette giver overblik over vores product backlog og hvad vi er igang med. Vi vil bruge trello som et værktøj i vores projekt ledelsesideologi SCRUM.

Google drev

Google drev giver nem og gratis adgang til værktøjer, så teamet kan arbejde i samme dokument når der skrives på rapporten.

Git og github

Git bruger vi til at føre versionsstyring af vores kode projekt, dette i samarbejde med vores favorit git klient, primært terminal men en gui klient kan ikke udelukkes.

Vi benytter github til at hoste vores projekt i en repository online, så vi kan tilgå koden uanset hvor vi befinder os.

Teamets roller

Da vi har valgt at benytte os af Scrum som projektstyringsmetode, er der nogle roller som skal klarlægges ved projektets start. Vi har som udgangspunkt tre forskellige roller som bliver opfyldt. Rollerne er som følgende:

Scrum Master

I dette projekt er Nikolaj Bang, blevet udpeget som scrum master. Hans ansvar i projektet vil bestå i at:

- Sørge for scrum meetings vil blive overholdt. Herunder skal teamets medlemmer hver i sær besvare spørgsmålene:
 - a. Hvad blev der opnået dagen forinden?
 - b. Hvad skal der arbejdes på i dag?
 - c. Er der nogle udfordringer ved dagens opgave?
- Derudover vil han have ansvaret for at teamets kanban board bliver holdt opdateret.

Product Owner

I forbindelse med Andreas' praktikforløb hos Synergi reklame- og webbureau, fik han, i samarbejde med hans daværende mentor, ideen til produktet der vil blive udarbejdet i dette projekt. Desværre kan mentoren til tider have for travlt til at kunne deltage som product owner. Derfor vil Andreas til tider agere product owner, da han løbende har kontakt med Synergi og derfor også kan stille spørgsmål omkring eventuelle krav til produktet.

For at vores projekt bedst muligt afspejler, at dette er et reelt produkt, er det vigtigt at Andreas derfor har styr på følgende:

- Hvad visionen er for produktet.
- Udvikling af User stories og deres acceptance kriterier.
- Fuldt overblik over product backlog og holder den opdateret.
- Give konstruktiv og objektiv feedback på resultater ved sprint reviews.

Team Members

Til sidst har vi rollen som team member. Dette er rollen som vi begge bliver tildelt som en del af udviklingsteamet. Som en del af teamet har vi alle ansvar for følgende:

- Udvikling af produktet.
- Deltage scrum planlægning, reviews og retrospectives.
- Deltage i scrum møder.
- Aktivt at bruge kanban board.

Derudover er der aftalt ansvar i forhold til udarbejdelse af rapport. Herunder:

- Begge udarbejder rapport løbende.
- Læse korrektur for hinanden

Product Backlog

Vi har i samarbejde med product owner udarbejdet vores product backlog. Denne backlog indeholder essentielle dele af systemet, som tilsammen kan realisere projektets helhed. På de user stories vi har sat på product backloggen, har de prioritet og en indikation for hvornår product owner kan forvente en user story er klar til release.

User stories

Herunder ses de user stories, som vi har arbejdet ud fra i de forskellige sprints.

De er udarbejdet med Andreas som product owner og er baseret på samtaler med Bo fra Synergi.

De er konstrueret ud fra en standard "user story template" som ser således ud:

"Som <bruger>, vil jeg <et mål> så jeg kan <en grund>".

Samtidig har alle user stories fået en business value af product owner. Business value er givet ud fra en værdi mellem 0-100, hvor 100 giver product owner mest værdi og 0 giver mindst.

1. Som supporter vil jeg kunne logge ind og se mine tickets så jeg har overblik over hvem der mangler min hjælp.
 - a. Acceptance kriterier:
 - Antal af tickets som afventer handling fra aktuelle bruger vises på forside.
 - b. Business value: 85
2. Som supporter vil jeg kunne skrive noter til tickets løbende så jeg har overblik over alle sagens tiltag.
 - a. Acceptance kriterier:
 - Noter tilføjes til den pågældende tickets historik.
 - Noter kan læses af alle supportere.
 - Noter gemmes i database.
 - b. Business value: 65
3. Som supporter vil jeg kunne ændre status på tickets løbende så jeg kan se hvor mange der mangler handling.
 - a. Acceptance kriterier:
 - Status skal ændres i hele systemet umiddelbart efter ændring er udført.
 - b. Business value: 35
4. Som supporter vil jeg kunne se alle support sager så jeg har overblik over hvor mange tickets vi har åbne.
 - a. Acceptance kriterier:
 - Alle tickets skal vises på en liste.
 - Det skal være muligt at ændre på sorteringen af tickets (eks. status).
 - Det skal være muligt at klikke på en ticket for at åbne den.
 - b. Business value: 100
5. Som supporter vil jeg kunne se en tickets historik så jeg har overblik over hvilke tiltag der tidligere er blevet gjort.

a. Acceptance kriterier:

- På en tickets historik skal supporter kunne se andre supporters tiltag og noter.
- Der skal være datostempel på alle handlinger i historikken.
- Der skal stå hvilken support bruger der har udført handlingen eller skrevet noten.

b. Business value: 45

6. Som supporter vil jeg kunne søge efter tickets så jeg hurtigt kan finde en support sag.

a. Acceptance kriterier:

- Der skal kunne søges på en ticket ud fra navn på kunde.
- Der skal kunne søges på en ticket ud fra kundens firmanavn.
- Der skal kunne søges på en ticket ud fra ticket Id.
- Der skal kunne søges på en ticket ud fra overskrift.

b. Business value: 60

7. Som supporter vil jeg kunne svare tilbage på supportsager inde i den pågældende ticket så jeg ikke lader kunder vente i uvished.

a. Acceptance kriterier:

- Der skal kunne skrives en tekst i et felt i den pågældende ticket som sendes via mail til kunden.
- Systemet skal automatisk indsætte en mail signatur som er gemt til bruger.
- I mailen der sendes til kunde skal ticket id være inkluderet i mailens emnefelt.

b. Business value: 80

Vores første version af product backloggen blev udarbejdet i samarbejde med product owner, og er en tabel som kan ses vedhæftet i bilag. Product backloggen vil kunne ændre sig i løbet af hele projektet, og vi vil derfor benytte os af Trello som product backlog. Trello kan bruges som et kanban board, som gør det super nemt at tilføje nye ting der skal laves og arrangere de eksisterende opgaver. Herunder er der en simplificeret version af product backloggen med de tidligere nævnte releases og business values, som er givet til hver user story i samarbejde med product owner. Det skal dog noteres at releases først blev tilføjet efter planning poker, som vil blive beskrevet i næste afsnit.

Nr.	Prioritet	Beskrivelse/business value	Release
1	2	Som supporter vil jeg kunne logge ind og se mine tickets så jeg har overblik over hvem der mangler min hjælp. Business value: 85	Begynder udvikling: sprint 1 Forventet release: Sprint 3
2	4	Som supporter vil jeg kunne skrive noter til tickets løbende så jeg har overblik over alle sagens tiltag. Business value: 65	Begynder udvikling: sprint 4 Forventet release: Sprint 5
3	7	Som supporter vil jeg kunne ændre status på tickets løbende så jeg kan se hvor mange der mangler handling. Business value: 35	Begynder udvikling: sprint 7 Forventet release: Sprint 7
4	1	Som supporter vil jeg kunne se alle support sager så jeg har overblik over hvor mange tickets vi har åbne. Business value: 100	Begynder udvikling: sprint 1 Forventet release: Sprint 1
5	6	Som supporter vil jeg kunne se en tickets historik så jeg har overblik over hvilke tiltag der tidligere er blevet gjort. Business value: 45	Begynder udvikling: sprint 6 Forventet release: Sprint 7
6	5	Som supporter vil jeg kunne søge efter tickets så jeg hurtigt kan finde en support sag. Business value: 60	Begynder udvikling: sprint 5 Forventet release: Sprint 6
7	3	Som supporter vil jeg kunne svare tilbage på supportsager inde i den pågældende ticket så	Begynder udvikling: sprint 3

		jeg ikke lader kunder vente i uvished. Business value: 80	Forventet release: Sprint 4
--	--	--	--------------------------------

Planning poker

Efter at have lavet vores product backlog og product owner havde været med til at give user stories deres prioriteter, ville vi gerne kunne lave et estimat af de forskellige user stories. Dette ville vi gerne lave fordi at det gav os mulighed for at udarbejde en mere nøjagtig tidsplan, og for at product owner kunne planlægge nogle realistiske releases.

Derfor valgte vi at benytte os af planning poker. Vi har tidligere haft god erfaring med planning poker og vi syntes at det giver en god mulighed, for at få snakket de forskellige user stories igennem. Således undgår man også eventuelle misforståelser, omkring hvordan de skal implementeres i den endelige løsning. Herunder kan man se et billede af det online planning poker vi benyttede os af, samt de forskellige user stories med deres estimering.

Active Stories	0	Completed Stories	7	All Stories	7	+ New	Edit
TITLE:				EST.:	TIME.:		
Som supporter vil jeg kunne svare tilbage på supportsager inde i den pågældende ticket så jeg ikke lader kunder vente i uvished.				13	00:03:24		
Som supporter vil jeg kunne søge efter tickets så jeg hurtigt kan finde en support sag.				8	00:03:58		
Som supporter vil jeg kunne se en tickets historik så jeg har overblik over hvilke tiltag der tidligere er blevet gjort.				8	00:02:46		
Som supporter vil jeg kunne se alle support sager så jeg har overblik over hvor mange tickets vi har åbne.				5	00:00:40		
Som supporter vil jeg kunne ændre status på tickets løbende så jeg kan se hvor mange der mangler handling				2	00:01:24		
Som supporter vil jeg kunne skrive noter til tickets løbende så jeg har overblik over alle sagens tiltag				5	00:00:19		
Som supporter vil jeg kunne logge ind og se mine tickets så jeg har overblik over hvem der mangler min hjælp.				13	00:01:17		

Med de estimerede story points på hver user story, kunne vi begynde at beregne hvornår den forventede release kunne sættes.

Vi har arbejdet sammen før og har derfor en god ide om at vi sammen, kan nå en velocity på 8 story points per sprint. Ud fra denne estimering kunne vi afsætte 8 story points per sprint og derfor give product owner, en god ide om hvornår han kunne forvente releases.

Prototype

Som det sidste inden vi gik i gang med at udvikle produktet, valgte vi at udarbejde nogle mock ups, så der var enighed om hvordan det skulle se ud. Disse mock ups er blevet udarbejdet af Andreas, som sad stand in for product owner og Nikolaj i rollen som udvikler team.

Vi er begge tilfredse med resultatet og det stemmer flot overens med den vision, som først var tiltænkt af Bo. Vi vil kort beskrive den vigtigste side i web applikationen og de tanker vi har gjort os derom:

SupportWebDesk

[Startside](#)
[Se Alle Tickets](#)
[Historik](#)
[Søg](#)
[Log ud](#)

Se Alle Tickets

Åbne tickets lige nu
23

Kritiske tickets
3

Status	Prioritet	Øverskrift	Anmoder	Ansvarshavende	Sidst opdateret
Åben	Høj	Hjælp mig	Jørgen Sørensen lvs	Agent 1	02-11-2017 13:37
Åben	Normal	Hjælp Jørgen	flans søn	Agent 1	02-11-2017 13:37
Åben	Kritisk	Word virker ikke	Person	Agent 2	02-11-2017 13:37
Åben	Normal	serveren er crashed	person	Agent 3	02-11-2017 13:37
Åben	Normal	Min browser lukker	person	Agent 2	02-11-2017 13:37
Åben	Normal	Lorte service	person	Agent 1	02-11-2017 13:37
Åben	Normal	Jeg har ikke signal	person	Agent 5	05-10-2017 13:37
Lukket	Normal	Intet problem	person	Agent 1	02-10-2017 13:37
Lukket	Normal	min ven jeg er sulten	person	Agent 4	01-11-2017 13:37
Lukket	Normal	Durum?	person	Agent 1	01-11-2017 13:37

Siden ovenfor er til visning af alle de support sager(herfra refereret til som "Tickets"), som ikke er blevet løst og hvor kunderne stadig mangler at få hjælp. Fra start af har det aldrig været design der var det vigtigste. Det er et program der skal bruges professionelt i erhverv, og det er derfor det praktiske der har første prioritet. Det er også grunden til opbygningen som den ser ud på vores mock up prototyper. Det er et meget let genkendeligt design som er let at gå til. Fælles for alle views, er en menu i venstre side, med en række fane punkter som gør det let at navigere til det ønskede view. I midten har man så det valgte view som man kan arbejde med. I viewet ovenfor er det eksempelvis alle de åbne tickets der bliver vist. De er hver i sær vist på en række i en tabel, med de mest essentielle info omkring hver ticket. Herfra kan du så navigere ind og se mere info om hver ticket ved at klikke på dem. Designet giver altså et hurtigt overblik over åbne tickets, og gør det let at navigere videre til andre views. Det er brugervenligt og super praktisk i en travl hverdag. Resten af vores views er designet ud fra de samme tanker og er vedhæftet i vores bilag¹².

¹² Se bilag 1.

Sprint 0 refleksion

Generelt for sprint 0 har vi kun brugt de "artefakter" som gav mening for os, og som vi følte vi skulle bruge for at etablere et godt fundament for opgaven. Som det første udarbejdede vi et vision statement i samarbejde med product owner, og det er nok en af de bedste beslutninger vi tog i sprint 0. Dette gjorde at vi fik en rigtig god forståelse for hvad det var product owner ønskede, og det fik også os til at stille spørgsmålstejn ved nogle af hans visioner, som gjorde at vi fik sat gang i en rigtig god dialog.

Efter at have udarbejdet et vision statement, havde vi en god ide om hvad der skulle laves. Ud fra dette kunne vi så internt åbne en dialog omkring hvorvidt dette var noget, vi kunne klare. For at danne os et overblik over hvad der kunne gå galt i sådan et projekt, besluttede vi os for at lave en risikoanalyse.

Risikoanalysen gjorde at vi var tvunget at sætte en finger på pulsen og vurdere hvilke risici, der var ved sådan en opgave. Vi kom frem til at de fleste af vores bekymringer gik på om hvorvidt vi havde de tekniske kompetencer, der skulle til for at udføre opgaven som vi selv, men også product owner ønskede. Det gode ved risikoanalysen var, at vi kom frem til at de fleste udfordringer der kunne opstå, var noget vi selv kunne gøre noget ved. Derfor valgte vi at tage den risiko, som der var størst chance for at vi kom ud for og forsøge at mitigere den.

Teamet besluttede sig for at lave et technical spike. På den måde fik vi kastet os ud i noget vi ikke havde prøvet før og gjorde det i et stadie af projektet, hvor det ikke ville have alt for stor impact, hvis vi kom til at bruge lidt mere tid end forventet. Det var også en god beslutning at lave testen i den forstand at, det gav os en masse selvtillid til at fortsætte med projektet. Vi havde allerede overkommet den første udfordring, før vi var gået rigtigt i gang. Det var også her at, vi første gang fik udbytte af Extreme programming, da vi udførte vores test ved hjælp af pair programming. Dette fungerede så godt for os at, vi måske vil bruge det i langt højere grad end forventet i selve udarbejdelsen af projektet. I dette tilfælde hjalp det os ekstremt meget at have to sæt øjne på kodningen, da det ikke var noget vi havde arbejdet med før.

Det var også i sprint 0 at, vi besluttede os for hvilken arkitektur og hvilke teknologier vi ville benytte os af. Arkitekturen var vi hurtigt enige om. Vi syntes begge at det var smart at være fremtids klar, hvis kunden ønskede at have en mobil applikation til support sager også. Derfor faldt valget på en Angular front end og så et API, som kunne kommunikere med vores backend. Med hensyn til teknologierne faldt valget på en backend, som vi havde arbejdet med før og en front end med Angular, som vi ikke kendte så meget til. Det var vigtigt for os at vi forsøgte at udfordre os selv og prøve nye teknologier. Når det så er sagt tror vi at det

var godt at, vi ikke lagde an til en større udfordring end vi kunne håndtere. Havde vi valgt nogle andre backend teknologier havde det måske flyttet fokus fra produktet/processen, og endt ud i at vi havde brugt mange ressourcer, på at udforske nye teknologier.

Med den fundamentale ide for produktet og de tekniske detaljer på plads, følte teamet at det var nødvendigt at klarlægge, hvem der ville påtage sig ansvaret i visse situationer. Derfor valgte vi at lave et stykke i rapporten, der beskrev hvilke roller vi ville påtage os, i forhold til Scrum. Det er ikke sikkert at disse roller vil blive brugt meget aktivt, men så er der taget stilling til det i tilfælde af at, vores product owner eksempelvis, beslutter sig for at han ikke vil deltage i projektet alligevel.

Som noget af det sidste i vores sprint 0, udarbejdede vi nogle user stories. Det gik utrolig godt, eftersom Andreas havde haft en god dialog, med den rigtige product owner. Derudover har de hjulpet os yderligere til at få en ide om hvordan programmet skal bruges, og hvilke ideer product owner har. Efter at have udarbejdet user stories, gik vi i gang med planning poker.

Planning poker er igennem uddannelsen blevet en værktøj, som vi sætter pris på og ikke undervurderer. I dette projekt gjorde det at, vi fik diskuteret hver enkelte sag og vi fik fjernet rigtig mange misforståelser, omkring hvad der var forventet for de forskellige user stories. Det gav også en enighed udviklerne imellem at få estimeret størrelsen af de forskellige user stories, da det åbnede dialog omkring den mere tekniske del af implementeringen.

Som det sidste i sprint 0, inden udviklingsfasen for alvor gik i gang, besluttede teamet sig for at udarbejde nogle visuelle prototyper, af hvordan det endelige resultat kunne se ud. Efter at prototyperne var blevet udarbejdet vidste vi at vi alle var enige om hvad det var vi skulle udvikle. I sprint 0 havde vi nu fået snakket om hvad product owner forventede af produktet, hvad det rent teknisk skulle opfylde, estimeret hvor langt tid det ville tage at udvikle og fastlagt hvordan det skulle se ud rent visuelt.

Alt i alt følte vi os klar til at gå i gang med projektet, og følte ikke at der var nogle aktiviteter, der kunne gøre os mere forberedt på opgaven, vi havde foran os.

Sprint 1

Sprint planning

I den første sprint går vi i gang med user story 4, da den har højeste prioritet på product backloggen. Dette er også den user story, som vil bidrage med mest business value til product owner.

Som team ønsker vi at give så meget business value til product owner som muligt hver sprint. Men for at være realistiske og ikke gabe over for meget, har vi estimeret at vi kan gennemføre 8 story points per sprint. Dette har vi baseret på tidligere erfaring.

Da user story 4 ikke er estimeret til mere end 5 story points begynder vi også udviklingen af user story 1, som har 2. prioritet på product backloggen.

For at have et overblik over ugen og opgaverne der skal laves, har Scrum master og team member lavet en plan for hvilke dage der skal laves hvad.

Ugeplan Sprint 1	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
Opaver:	Scrum møde Oprette sprint backlog. Lave user story 4 og 1 om til tasks. Oprette projektet hente de tilføjelser som vi vil arbejde med.	Udarbejdelse af den første task: Heriblandt: Database og Backend	Scrum møde Videre arbejde med tasks fra user story 4 heriblandt: Backend og API	Færdig udarbejdelse af tasks tilhørende user story 4. heriblandt: API og Angular front end	Scrum møde Påbegynde udarbejdelse af tasks fra user story 1. Sprint review + Retrospective

Løbende testing

Som man kan se i uge planlægningen ovenfor, er der ikke skrevet noget ind omkring tests. Dermed ikke sagt at der ikke er taget højde for det. Testing vil ikke være en visuelt planlagt aktivitet i vores sprint planning, men derimod være en selvfølge. For hver task der bliver

udarbejdet, vil der blive opstillet nogle test cases, så vi får testet vores kode løbende. De opstillede tests vil så blive vist i afsnittet omkring tekniske detaljer.

Sprint backlog

Som sprint backlog bruger vi Trello, for at gøre det nemt for os at rykke de forskellige tasks rundt, når det er nødvendigt. På sprint backloggen har vi de user stories, som der skal arbejdes på i den pågældende sprint. Nedenfor inkluderer vi den første version af vores sprint backlog i en simplificeret version.

User stories	Tasks	In progress	Done
US 4: Som supporter vil jeg kunne se alle support sager så jeg har overblik over hvor mange tickets vi har åbne.	Angular service som kan benytte sig af API endpoint til udtrækning af data. Design angular komponent til visning af tickets. Code first Design database til tickets. Oprette API endpoint til at trække data ud af database.		
US 1: Som supporter vil jeg kunne logge ind og se mine tickets så jeg har overblik over hvem der mangler min hjælp.	Oprette brugersystem. Design Angular komponent til login. Angular component for bruger forside. Oprettet et API endpoint til at trække tickets ud til bestemt bruger. designer bruger authorization til controllere.		

Da vi delte vores user stories ud i tasks, valgte vi at holde hver task i en størrelse, som gjorde at de ikke var for små, men det var også vigtigt for os at man kunne nå en task om dagen, så de skulle heller ikke være for store. Hver task i vores sprint backlog vil cirka kunne udføres på en dag.

Derudover har vi i teamet aftalt at der maks må være 3 tasks på “doing”. Dette betyder at hvis vi har gang i 3 tasks, skal vi holde fokus på at få dem færdigarbejdet, før vi går i gang med nye tasks.

Scrum møder

En vigtig del af scrum er at holde scrum møder, hvor der bliver snakket om hvad man har lavet, hvad der skal laves og om teamet står overfor nogle udfordringer, der kan gøre deres arbejdsgang svær.

Da vi kun er to mand i teamet og har god kommunikation mellem os, har vi valgt at holde scrum møder tre gange om ugen.

Mandag holder vi møde for at snakke om hvad der skal laves og om vi begge er med på, hvordan vi skal gribe opgaverne an. Næste sprint møde bliver onsdag for at holde et “opdaterings” møde midt i sprinten. Dette gøres blandt andet også for at holde hinanden opmærksomme på om der er noget, vi skal holde øje med resten af sprinten.

Det sidste scrum møde bliver holdt fredag, for at være sikre på at, vi når de sidste ting som vi har planlagt at få færdige i den pågældende sprint.

Opsummering af scrum møder

Mandag

På vores første scrum møde diskuterede vi hvordan vi skulle gribe opgaven an. Vi besluttede os for at holde os til vores sprint planlægnings tabel, og udarbejde sprint backlog og uddele user stories til tasks sammen. Herefter vidste vi der meget der skulle laves og da vi begge gerne ville være med til at sætte projektet op og sparre med hinanden valgte vi at, benytte os af en arbejdsform fra extreme programming kaldet “pair programming”.

Onsdag

På andet scrum møde havde vi ikke så meget fokus på at besvare spørgsmålet, om hvad vi hver i sær havde lavet. Grundet godt samarbejde med pair programming den første dag, valgte vi nemlig at gå videre med samme fremgangsmåde om tirsdagen. Derfor var scrum mødet holdt med fokus på hvad der fremadrettet skulle laves. Sammen ville vi arbejde på backend og API resten af dagen og fortsat følge vores sprint planlægning. På daværende tidspunkt var vi ikke stødt ind i det store problemer.

Fredag

På sidste scrum møde holdte vi en lille opsummering, på hvad vi sammen havde fået lavet for at gå det igennem en ekstra gang. Vi havde arbejdet ved hjælp af pair programming igen og vi er begge overrasket over hvor godt det fungerer. Til sidst snakkede vi om hvordan resten af dagen skulle forløbe. Vi aftalte at udarbejde en enkelt task for næste user story, og efterfølgende holde sprint review og retrospective for sprint 1.

Sprint review

Som noget af det sidste i vores sprint, holdte vi et møde med product owner og snakkede med ham om hvad vi havde nået. Heriblandt snakkede vi om hvorvidt, vi havde nået de user stories, som han havde prioriteret højt fra product backloggen, og om vi havde overholdt sprint planlægningen.

Den første sprint er forløbet som vi kunne have håbet. Vi har leveret den business value som product owner forventede. Vi gav product owner muligheden for at afprøve det arbejde, der er blevet udført indtil videre. Product owner var tilfreds med det foreløbige resultat, og havde kun få ændringer til udseendet af produktet.

Det første ønske fra product owner var meget simpel. Da vi havde planlagt at de forskellige tickets skulle have en "prioritet" i programmet, så supporterne kunne se hvilke sager der hastede, fik vi ikke snakke om hvilke typer af prioriteter der skulle være. I produktionen havde teamet taget en beslutning om at give dem tre forskellige værdier; lav, mellem og høj. Product owner var enig med os i de tre værdier, men manglede en ekstra værdi for hastesager. Derfor ønskede han en ticket prioritet som hed "kritisk" også.

Det sidste ønske var en tilføjelse til designet, som ikke var bemærket ved forevisning af mockup prototyperne. Når man har navigeret til en bestemt ticket og kan læse alle detaljer herom, ville product owner gerne have en mulighed for at navigere tilbage til det fulde ticket view igen. Der blevet diskuteret frem og tilbage hvordan man kunne gøre dette og teamet forelagde ham en anden ide også. Når man havde åbnet en ticket ville den blive tilføjet som en fane i navigationsbaren ude til venstre. Dette ville gøre at man nemt kunne finde tilbage til de tickets som man sad og arbejdede med, selvom man eksempelvis måtte ind og bruge søgefunktionen til at finde noget historik for en tidligere ticket. På den måde ville man altså primært bruge navigationsbaren i venstre side af programmet til at navigere rundt med. Dette ville så også gøre sig gældende, til det ønske som product owner har.

Det blev aftalt at begge muligheder skulle implementeres, for at se hvordan det fungerer i praksis og som product owner selv sagde; "Så kan tilbageknappen jo altid bare fjernes igen". Hertil skal det tilføjes at product owner er indforstået med at der så potentielt vil blive brugt ressourcer på noget som ikke nødvendigvis udkommer i produktion.

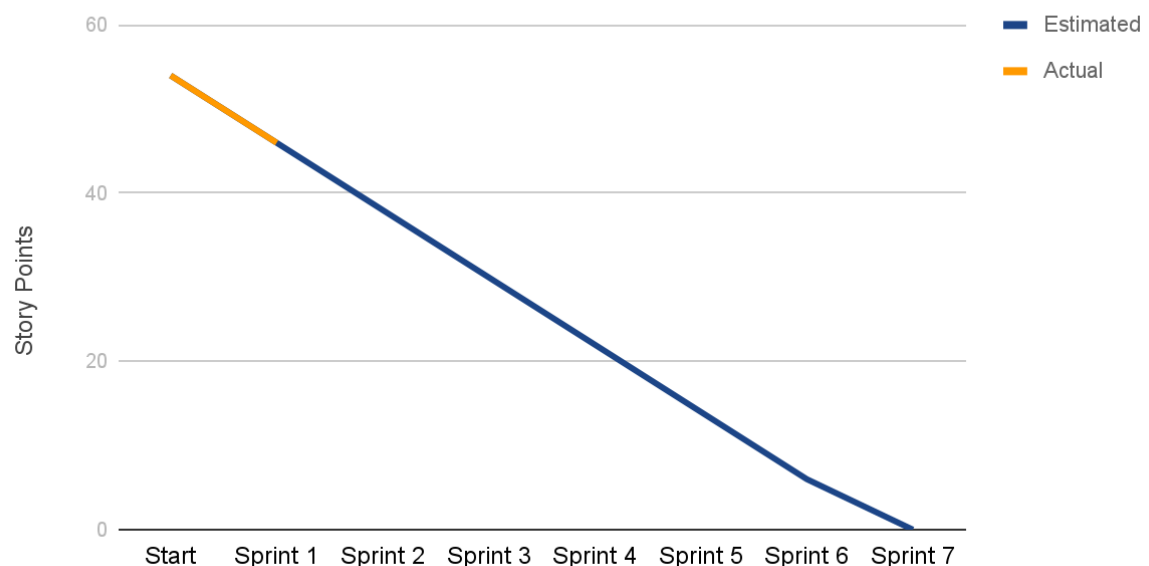
Alt i alt hvor product owner godt tilfreds med første sprint og syntes at det var fint at teamet tog nogle beslutninger undervejs, som eksempelvis at bruge lav, mellem og høj som prioriteter selvom det ikke var blevet aftalt med ham. Dette er også en følge af at product owner selv har travlt og derfor ikke kan stå til rådighed i løbet af hele sprinten.

Burndown chart

Ud fra grafen kan vi se at teamet følger godt med. Teamet når de estimerede 8 story points per uge. Selvom user story 4 var 5 story points, havde vi påbegyndt arbejdet med nogle tasks til sprint 2, for at kunne fylde tiden ud. Projektet er stadig i dets tidligere stadier, så vi kan slet ikke på nuværende tidspunkt konkludere om teamet, kan nå at blive færdige. Følger teamet den nuværende pace, fra sprint 1, vil teamet nå i mål med et fint resultat og aflevere produktet til tiden.

Burndown Chart

End of Sprint 1



Sprint Retrospective

Efter denne sprint diskuterede teamet kort omkring hvad der gik godt, hvad vi har lært, hvad der kunne gøres anderledes og hvad der er gået knap så godt i vores sprint.

Teamet har haft rigtig god kommunikation, og teamet har blandt andet benyttet sig af pair programming til at udvikle programmet. Pair programming har vist sig at være et godt værktøj, dog skal der skiftes lidt imellem hvem der sidder foran tastaturet.

Vores scrum møder har været fine, det har hjulpet teamet med at holde overblik over sprinten.

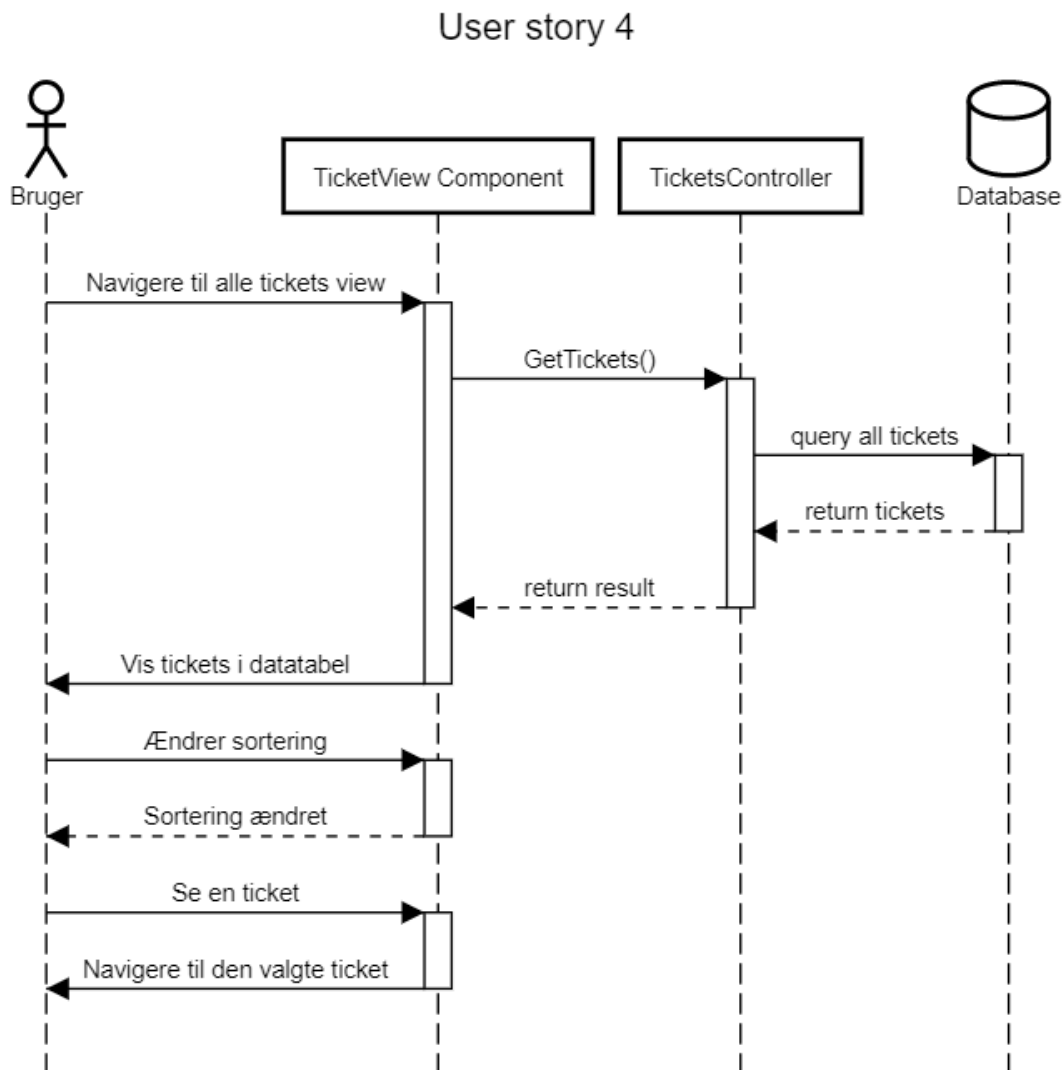
Teamet har i user story til tasks processen, ikke fået reflekteret user stories acceptance kriterierne grundigt nok. Dette vil vi i fremtidige sprints forbedre, da dette er essensen af hvordan arbejdet skal udføres og hvilken retning.

Dette har resulteret i mere uforudset arbejde, der har dog været rigeligt med tid, så dette er også blevet udført i denne sprint.

Tekniske detaljer

Da denne rapport er en del af datamatiker hovedopgaven, har vi valgt også at inkludere et afsnit med tekniske detaljer. Dette vil ligge i slutningen af hver sprint.

User story 4 indeholder fire tasks, hvor vi efterfølgende har opdaget at, user story 4's acceptance kriterierne ikke bliver reflekteret i disse tasks. Derfor inkludere vi et kort afsnit, hvor vi redegør for det arbejde vi har lavet, for at opfylde PO's acceptance kriterier. For at visualisere user story 4, har vi arbejdet ud fra vores mockup "se alle tickets" og lavet dette diagram:



Diagrammet er en sammenlægning af et sekvens diagram og et system sekvens diagram. Vi har valgt at lave det sådan for at begrænse mængden af billeder i selve rapporten. Vi syntes at dette fint visuelt illustrerer hvad koden for den pågældende user story gør og vil vise diagrammer på samme måde i de kommende sprints.

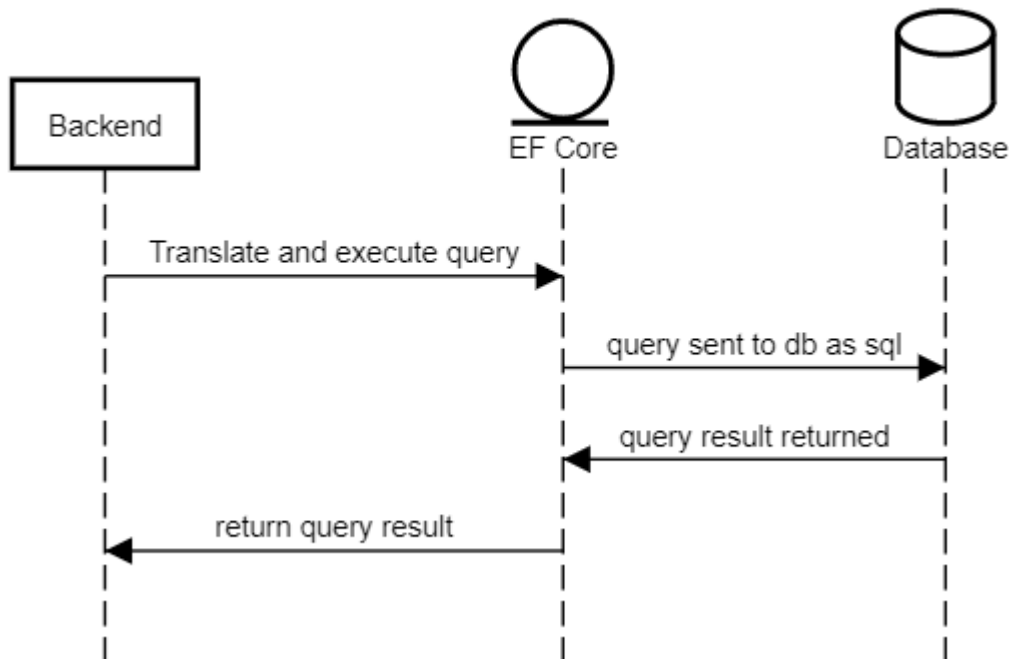
Code First Design database til tickets

Vi benytter tilgangen til database design, hvor vi først koder klasserne og får deres sammenhæng i programmet til at virke. Dette sikrer os fuld kontrol over vores kodebase, så vi ikke bliver afhængige af autogenerated kode. Derudover giver Entity framework core tools¹³ mulighed for at generere disse migrations, det opretter en dynamik med database versionskontrol, samt at koden definerer databasen. Vi har valgt denne fremgangsmåde da vi hellere vil lægge logikken i programmet end i databasen, databasen skal udelukkende være storage. Dette tillader at vi kan hurtigt sætte vores program op sammen med en frisk

¹³ <https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.Tools/>

database, da entity framework core genererer SQL'en, som definere hvordan databasen skal se ud, ud fra vores migrations filer.

How EF Core works



Migrations

For at lave denne database, benytter vi Entity Framework core's Migration toolings¹⁴. Dette værktøj er et "dotnet" Command-line interface, som finder database contexts man har defineret i sit projekt. Den kører efterfølgende metoden "OnModelCreating" for at registrere om der skal laves en migration fil.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);
    modelBuilder.Entity<Ticket>().ToTable("Tickets");
}
```

Metoden benytter et modelbuilder objekt, hvorpå man fortæller hvilke klasser er ens Model klasser, så man kan mappe den til en database tabel. Denne CLI tool opretter et versions snapshot af de model klasser, man definere i "OnModelCreating", hvorved denne kan registrere dette og inkludere dem i en migration. Sker der ændringer i ens model klasser, kan man benytte CLI værktøjet til at lave en migration, som reflektere disse ændringer. En migration er en klasse, som benyttes til at udtrykke et stadie af en database version, denne klasse indeholder en beskrivelse af ændringerne, i let læselig C# syntax. Indholdet i en migration er en serie af statements, som Entity Framework Core kan oversætte til den

¹⁴ <https://docs.microsoft.com/en-us/aspnet/core/data/ef-mvc/migrations>

pågældende SQL database. Disse migrations indgår i en "migrationhistory", som fuldt ud beskriver hvordan ens database, er blevet genereret. Herunder ses en migration, denne har en Up og en Down metode, hvor Up bliver kørt når migrationen skal køres på databasen, Down køres når migrationen skal fjernes fra databasen.

```
public partial class emailsignatureadd : Migration
{
    8 references | Bang0123, 27 days ago | 1 author, 1 change | 0 exceptions
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.AddColumn<string>(
            name: "EmailSignature",
            table: "AspNetUsers",
            type: "nvarchar(max)",
            nullable: true);
    }

    8 references | Bang0123, 27 days ago | 1 author, 1 change | 0 exceptions
    protected override void Down(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.DropColumn(
            name: "EmailSignature",
            table: "AspNetUsers");
    }
}
```

Oprette API endpoint til at trække data ud af database

Visual studio gør det meget ligetil at scaffolde en controller, hvori man kan oprette disse endpoints. Her oprettede vi 3 endpoints, et endpoint der returnerer alle tickets, et endpoint der returnerer et tal på, hvor mange tickets med prioritet kritisk der er og et der returnerer, hvor mange tickets der har status åbne.

```
/// <summary>
/// GET: api/Tickets
/// Returns all tickets
/// sorted by updatedat dsc
/// </summary>
/// <returns></returns>
[HttpGet]
0 references | Bang0123, 3 days ago | 1 author, 3 changes | 0 requests | 0 exceptions
public IList<TicketViewModel> GetTickets()...
/// <summary>
/// GET: api/Tickets/openamount
/// Returns amount of tickets that are open
/// </summary>
/// <returns>integer</returns>
[HttpGet("openamount")]
0 references | Bang0123, 3 days ago | 1 author, 1 change | 0 requests | 0 exceptions
public async Task<IActionResult> GetOpenTicketsAmount()...
/// <summary>
/// GET: api/Tickets/criticalamount
/// Returns amount of tickets that are critical
/// </summary>
/// <returns>integer</returns>
[HttpGet("criticalamount")]
0 references | Bang0123, 3 days ago | 1 author, 1 change | 0 requests | 0 exceptions
public async Task<IActionResult> GetCriticalTicketAmount()...
```

Design angular komponent til visning af tickets

Efter at have researchet angular 4, igennem videoer på youtube og andre former for dokumentation. Kunne teamet hurtigt sætte sig ind i den template, der var genereret sammen med ASP.NET core projektet. Her gik teamet hurtigt i gang med at lave et view komponent, hvor man kan se alle tickets i en datatabel. Der blev taget udgangspunkt i vores mockup view med alle tickets, hvor der også er overblik over hvor mange kritiske og åbne tickets der er. Sidst i det overordnede afsnit, Tekniske detaljer vil der være et billede.

Angular service som kan benytte sig af API endpoint

For at vise noget data i de view komponenter teamet har lavet, har vi brugt for at centralisere kommunikationen med web api'et, dette gøres blandt andet i en angular service. Dette gøres ved hjælp af angular http modul, som gør det muligt at kalde http kald mod et endpoint.

Arbejdet for at opnå acceptance kriterierne

Teamet blev ved scrum mødet om fredagen, opmærksomme på at se om alle acceptance kriterier var opfyldt. User story 4's acceptance kriterier:

- *Alle tickets skal vises på en liste.*
- *Det skal være muligt at ændre på sorteringen af tickets (eks. status).*
- *Det skal være muligt at klikke på en ticket for at åbne den.*

Teamet kan dermed konkludere at de acceptance kriterier, ikke ordentligt er blevet reflekteret i de tasks der er blevet opstillet for sprint 1. Vi manglede derfor et view komponent der skulle vise en ticket, samt mulighed for at ændre i data tabellens sortering. Teamet gik i krig med at designe et ticket view komponent, og lave et link fra en række til et ticket view med den ticket i.

Derefter kom den lidt mere avancerede opgave, at kunne sortere i viewet der viser alle tickets i en datatabel. Her blev vi nødt til at opgradere vores angular app til version 5.0.1, da denne version af angular understøtter google's material themes, Material Angular¹⁵. Material Angular har allerede store kode biblioteker til dets komponenter, hvor vi har valgt at bruge Material table¹⁶. Denne type tabel giver mulighed for at benytte kode bibliotekets sorterings funktioner, som gør at dette hurtigt og nemt blev løst.

¹⁵ <https://material.angular.io/>

¹⁶ <https://material.angular.io/components/table/examples>

The screenshot displays a web application for a support desk. On the left is a dark sidebar with navigation links: 'Startside', 'Tickets' (highlighted), 'Ticket', and 'Logout'. The main content area is titled 'Se Alle Tickets' and shows a summary of 'Åbne tickets Lige nu' (2) and 'Kritiske tickets' (1). Below this is a table of tickets with columns for Status, Prioritet, Overskrift, Anmoder, Agent, and Sidst opdateret. Two tickets are listed, both with status 'Åben' and priority 'Normal'. The first ticket has the subject 'mYPwIT' and the second 'PA8PzF'. Each row has a 'Se Ticket' button. Below the table, the 'Ticket' details for 'PA8PzF' are shown. It includes the ticket ID (10), status (Åben), and priority (Normal). There is a button 'Gå til seneste svar i tråden'. The 'Anmoder' section shows 'admin@gmail.com' as both the requester and the responder. The 'Beskrivelse' section contains the text 'nLfhszMSBk' and a large text area for the description. On the right, there are radio buttons for 'Intern' (selected) and 'Ekstern', and a 'Send svar' button.

Status ↓	Prioritet	Overskrift	Anmoder	Agent	Sidst opdateret	Se Ticket
Åben	Normal	mYPwIT	admin@gmail.com	admin@gmail.com	16:13 15-11-2017	Se Ticket
Åben	Normal	PA8PzF	admin@gmail.com	admin@gmail.com	16:13 15-11-2017	Se Ticket

Ticket

PA8PzF

Ticket Id: 10
Status: Åben
Prioritet: Normal

[Gå til seneste svar i tråden](#)

Anmoder:
admin@gmail.com

Ansvarshavende:
admin@gmail.com

Ticket lavet:
16:13 15-11-2017

Ticket sidst opdateret:
16:13 15-11-2017

Beskrivelse
nLfhszMSBk

☒ Intern
☐ Ekstern

[Send svar](#)

Testing

I dette afsnit med testing vil der blive forklaret lidt, om den måde vi har lavet automatiske tests i angular app'en. Vi har primært lavet små unit tests i denne sprint, der vil dog ikke været lavet nogen automatiske acceptance tests endnu, men der vil blive opstillet nogle stykker.

Unit tests

For tests på unit eller integrations niveau, benytter vi karma¹⁷ som test runner og jasmine¹⁸ til at opstille test suites og tjekke expectations.

Her har vi lavet nogle tests af vores template, hvor vi tester om det er muligt at manipulere bagvedliggende variabler, og få vist et resultat der fortæller at vi kan. Herunder ses en test case, hvor opentickets bliver sat til et bestemt tal. Derefter søger dokument objektet efter ændringer, hvorefter den bliver compiled så det er nemmere at søge i, med en queryselector. Efterfølgende bliver der sat en expectation op, som er den der bestemmer en tests udfald.

¹⁷ <https://karma-runner.github.io/2.0/index.html>

¹⁸ <https://jasmine.github.io/>

```
it('should render text for open tickets and display amount', () => {
  component.openTickets = 69;
  fixture.detectChanges();
  const compiled = fixture.debugElement.nativeElement;
  expect(compiled.querySelector('div.row div[name="openticks"] h3').innerHTML)
    .toContain('Åbne tickets Lige nu');
  expect(compiled.querySelector('div.row div[name="openticks"] p').innerHTML)
    .toEqual('69');
});
```

For at køre disse tests benytter vi CLI kommandoen “ng test” for unit/integration tests.

Herunder kan ses resultatet af de 11 unit tests, der er lavet til componenterne.

```
11 specs, 0 failures

SingleTicketViewComponent
  should create the component
  should render h1 `Ticket`
  should render h1 `Ingen ticket valgt` if no ticket in dataservice
  should render a test ticket with messages
  should render a test ticket

TicketViewComponent
  should create the component
  should render h1 `Se Alle Tickets`
  should render text for open tickets and display amount
  should render text for critical tickets and display amount
  should render text for loading
  should render one row in datatable
```

Acceptance tests

Vi planlægger at udføre automatiske tests, som benytter protractor¹⁹ og en instans af selenium webdriver. Protractor kan køre tests og se det som om vi selv så det igennem en webbrowser. Jasmine står for at opsætte test suites, der står for at tjekke de expectations man stiller op. I disse tests er der ikke noget mocked eller fake, her tester vi black box med forventninger om at systemet som helhed fungerer. Alt i alt er denne test syntax fra jasmine frameworket, nem at forstå, så det giver god mulighed for at testene får dækket de kriterier, som der er tiltænkt.

Vi har opstillet følgende tests til senere implementation:

- Ticketview should Login and navigate to ticket and not see one
- Ticketview should navigate to tickets and see h1
- Ticketview should see tickets in All tickets datatable
- Ticketview should see whole ticket
- Allticketsview should see open tickets
- Allticketsview should see critical tickets
- Allticketsview should see tickets in All tickets datatable
- Allticketsview should change filter of tickets and back

¹⁹ <http://www.protractortest.org/#/>

Sprint 2

Sprint planning

I denne sprint går teamet videre med user story 1, da den har prioritet lige efter user story 4 på product backloggen.

User story 1 er estimeret til 13 story points, hvor vi har påbegyndt udviklingen af en enkelt task i sprint 1. Vi regner med at denne User story også opfylder den estimerede tid, dog af tidligere erfaring, kan en user story der indeholder et brugersystem, enten trække ud eller være nemt og hurtigt at implementere.

For at have et overblik over ugen og opgaverne der skal laves, har Scrum master og teamet lavet en plan for hvad der skal laves de efterfølgende dage.

Ugeplan Sprint 2	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
Opaver:	Scrum møde Organisere sprint backlog. Påbegynde user story 1's tasks.	Udarbejdelse af tasks: Heriblandt: Brugersystem og Authorization	Scrum møde Videre arbejde med tasks fra user story 1 heriblandt: Backend og API	Færdig udarbejdelse af tasks tilhørende user story 1. heriblandt: Angular Startside og frontend login	Scrum møde Påbegynde udarbejdelse af tasks fra user story 1. Sprint review + Retrospective

Sprint Backlog

På sprint backloggen har vi de user stories som der skal arbejdes på i den pågældende sprint. Vi opdelte denne sprints user story i tasks, i sidste sprint, så vi kan derfor hurtigt begynde med at arbejdet. Tasken "Oprette brugersystem" blev påbegyndt i sidste sprint, for at undgå redundant dobbeltarbejde og lægge et fundament for hvordan brugere bliver brugt i systemet. Nedenfor inkluderer vi sprint 2's version af vores sprint backlog i et simplificeret format.

User stories	Tasks	In progress	Done
US 1: Som supporter vil jeg kunne logge ind og se mine tickets så jeg har overblik over hvem der mangler min hjælp.	Design Angular komponent til login. Angular component for bruger forside. Oprettet et API endpoint til at trække tickets ud til bestemt bruger. designer bruger authorization til controllere.	Oprette brugersystem.	

Scrum møder

Teamet syntes at scrum møderne fungerede, som de skulle i sidste sprint, og derfor fortsætter vi som tidligere planlagt med at afholde dem på samme måde.

Opsummering af Scrum møder

Mandag

Mandag morgen blev et kort scrum møde. Da vi kun er to i teamet og vi i sidste uge benyttede os af pair programming, vidste vi begge godt, hvad der var blevet arbejdet på. Fokus blev derimod på hvad vi skulle lave på dagen, og der var massere at gå i gang med grundet starten af en ny sprint. Vi besluttede os for at starte med en sprint planlægning som i sidste sprint og vil forsøge at holde os til denne så vidt muligt.

Onsdag

Midtvejs i vores sprint afholdte vi endnu et planlagt scrum møde, hvor vi kort opsummerede hvad vi havde nået mandag og tirsdag. Denne gang var det en god ting at få gennemgået kort hvad der var blevet lavet, da vi havde brugt tid på individuelt arbejde også. Efter opsummering blev vi enige om, at det har fungeret godt med individuelt arbejde og det bidrager også til at vi kan variere vores arbejdsform, fra pair programming til individuelt opgaver en gang imellem.

Fredag

På vores dette scrum møde vidste vi begge godt hvad vi hver i sær havde arbejdet med. Torsdag gik vi tilbage til pair programming grundet tekniske udfordringer i koden. I samarbejde fik vi løst problemerne og derfor lagde vi fokus på hvad vi skulle lave på denne

sidste dag af sprinten i stedet. Da vi snakkede om hvilke udfordringer der kunne være for dagens opgaver, kunne Andreas fortælle at sprint review kunne blive en udfordring, da product owner måske ikke får tid til at deltage. Derfor valgte vi at benytte Andreas som stand in for product owner og eventuelt benytte os af nogle bruger test, i samarbejde med andre som kunne komme til at være "slutbrugere" af Support Web Desk.

Sprint Review

Som i sidste sprint ville vi gerne afslutte ved at få product owner til at se og afprøve de ting, der var blevet udviklet på produktet i ugens løb. I denne sprint var det dog en udfordring, da product owner ikke har haft muligheden for at stå til rådighed for sådan et event. Som aftalt på scrum mødet tidligere på dagen, har gruppen derfor valgt at lade Andreas sidde standin for product owner og forsøge at give objektiv feedback.

Dette er selvfølgelig svært da alle tilstedeværende er med til at udvikle på produktet, men det gjorde alligevel at vi fik gennemgået brugerfladen, på en anden måde og fik snakket om løsningen som den så ud, men også hvordan det ellers kunne have været gjort.

I løbet af sprinten blev der kommunikeret med den rigtige product owner, omkring muligheden for log ind og log ud. Der blev snakket kort omkring hvordan det skulle se ud og ønsket fra product owners side var, at log ud skulle ske i venstre side af skærmen i navigationsbaren. Derfor kunne Andreas i sprint reviewet også videregive denne feedback til teamet, som så senere kunne inkludere log ud i navigationsbaren, ud fra product owners ønske.

Derudover kunne Andreas som product owner se at de ting, som var blevet drøftet til sidste sprint review var blevet tilføjet som teamet havde lovet. Disse detaljer blev også kommunikeret til den rigtige product owner. At arbejde ud fra product owners feedback på den måde gør også at, man inkluderer ham og giver ham følelsen af at han er med til at forme produktet og hjælpe projektet i en god retning.

Brugertest

Da det er svært at give feedback på et produkt, som man selv har været med til at producere og grundet en product owner som har travlt, har vi valgt som et ekstra led i vores sprint review at afvikle nogle bruger tests. Disse bruger tests vil blive udført af andre IT-folk, som i sidste ende kunne være potentielle slutbrugere. Hvis disse tests skulle vise sig at være en

god oplevelse med feedback og god kommunikation vil vi i fremtiden afholde en sådan test efter hvert andet sprint.

Som tidligere beskrevet, blandt andet til vores mock up prototyper, var det vigtigt for os at designet er nemt at gå til og super brugervenligt. Derfor har vi valgt at vores bruger tests skal afholdes på følgende måde.

Vores testperson vil sidde med programmet åbent foran sig. Han vil herefter blive stillet nogle opgaver som han skal løse. Disse opgaver vil opfordre testpersonen til at navigere rundt i applikationen for at finde bestemt information. Et eksempel på en opgave som kan blive stillet: *“Som supporter skal du nu finde detaljeret information omkring ticket X”*.

Teamet vil så overvære hvordan testpersonen klarer sig og hvor hurtigt opgaven bliver løst. Hvis opgaven skulle vise sig at være svær og testpersonen ikke kan finde en løsning vil de blive præsenteret for den korrekte måde at gøre det på, og herefter blive adspurgt omkring hvad der kunne gøres bedre.

Vi vil kun inkludere en enkelt testperson til hver bruger test. Dette i sig selv giver ikke varieret feedback, men for at løse dette vil vi ikke lade den samme testperson tage testen flere gange.

Testen til denne sprint var hurtigt overstået. Dette kan der være to grunde til. Testpersonen var hurtig til at løse opgaverne og det vidner om en brugerflade som er let at gå til. Den anden grund er selvfølgelig også en stor faktor. Da programmet stadig er i et meget tidligt stadie er der ikke meget at navigerer rundt i endnu. Den vigtigste del for os var dog stadig at få noget konstruktiv feedback.

Efter testen snakkede vi med testpersonen omkring programmet. Han syntes at opgaverne var gode, da de fik ham rundt i programmet og det var nemt at finde den information man gerne ville frem til. Han havde ikke de store tilføjelser, men som han selv påpegede, så er det et program som er i et tidligt udviklingsstadie.

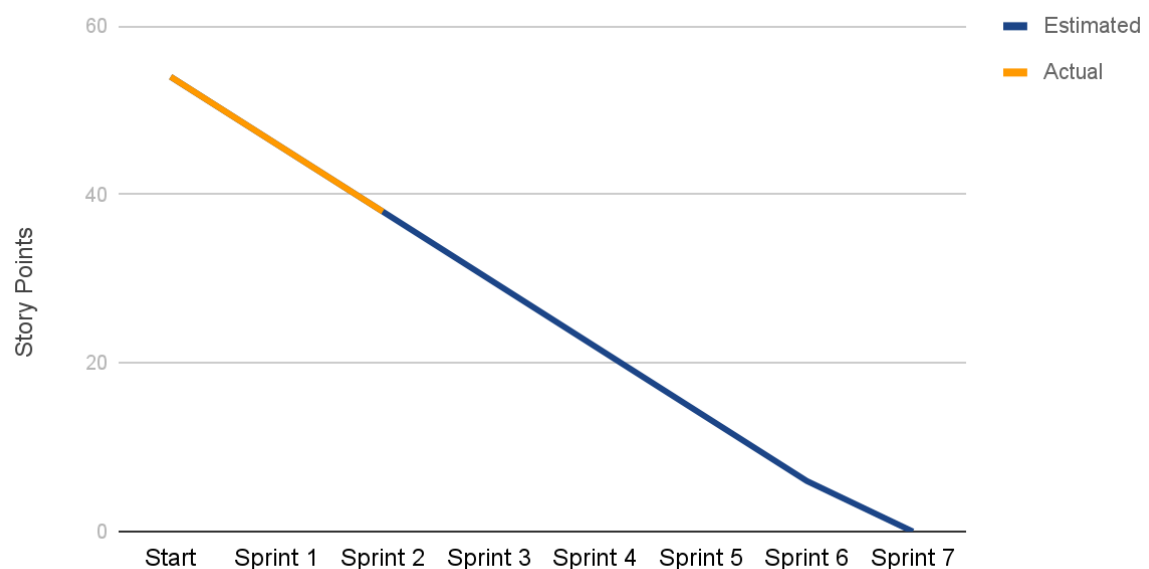
Vi føler at testen var en success, da vi fik bekræftet at det som vi havde produceret indtil videre var i den rigtige retning. Derudover kan vi se at inkludering af brugere og product owner er med til at holde os på tæerne og kan i sidste ende være med til at øge kvaliteten af produktet.

Burndown Chart

Teamet når de 8 story points per uge, og med user story 1 og 4 færdige, følger teamet en god pace. Projektet er ved at forlade de tidlige stadier, og har på nuværende tidspunkt fået lagt fundamentet, der baner vejen for at fuldføre projektet. Følger teamet den nuværende pace, fra sprint 2, vil teamet levere et fint resultat og aflevere produktet til tiden.

Burndown Chart

End of Sprint 2

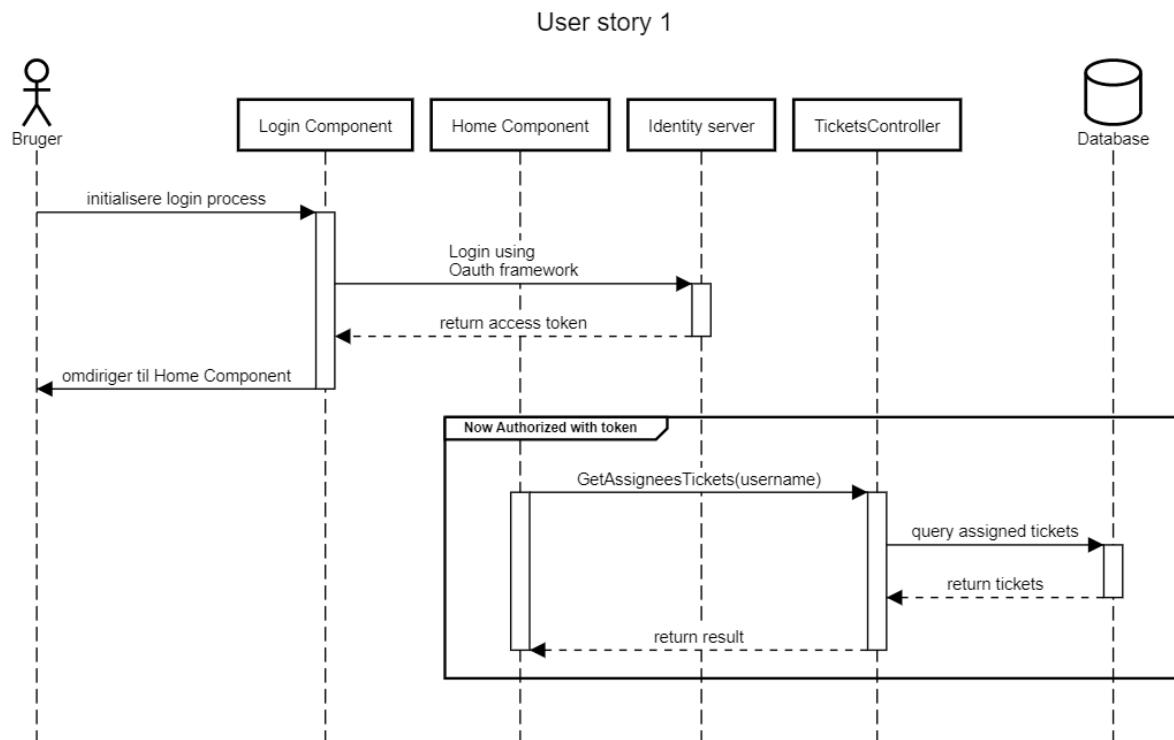


Sprint Retrospective

Teamet har fortsat haft god kommunikation, og teamet har fortsat benyttet sig af pair programming til at udvikle programmet, derudover har vi også arbejdet individuelt, dette har fungeret godt. Det var også fint at arbejde hver for sig, da vi så kunne sammenligne arbejdet med det vi har fået lavet når vi benyttede os af pair programming. Til scrum møderne har der blandt andet været lidt forklaring, da der har været lidt individuelt arbejde. På den måde har scrum møderne også fungeret som en statusopdatering har hjulpet teamet med at holde overblik over sprinten.

Tekniske detaljer

User story 1 består af fem tasks, disse tasks har reflekteret PO's acceptance kriterier fint, så arbejdet i sprinten har været håndterbart. User story 1 starter med at brugeren bliver mødt med en login side, hvorefter der sker et handlingsforløb for at nå til vores mock-up "Startside", dette har vi visualiseret med et sekvensdiagram:



Oprette brugersystem

Til at opnå dette benytter vi identitets frameworket identityserver4²⁰, dette framework bygger ovenpå ASP.NET's identity framework, bare for .NET core 2.0. Det krævede lidt at finde ud af lige præcis hvordan det kunne løse vores problem, men vi fik løst det hen over sprinten. Til identityserver4 kræves der at opsætte en masse konfigurations parametre. Dette er nødvendigt da identityserver4's identityserver middleware process bliver tilføjet til request pipelinen.

Design bruger authorization til controllere

Denne task blev lidt anderledes, da identityserver4 allerede er et komplet system til at authorization. Det eneste der skulle tilføjes til controllerne, var dette attribut tag. Tagget gør at identity frameworket skal foretaget de nødvendige authorizations tjek når en bruger henvender sig efter en ressource.

```
// Authorization policy for this API.  
[Authorize(AuthenticationSchemes = IdentityServerAuthenticationDefaults.AuthenticationScheme, Policy = "Access Resources")]  
1 reference | Bang0123, 5 days ago | 1 author, 6 changes  
public class TicketsController : Controller
```

Dette authorization schema er defineret i de konfigurations parametre, man giver identityserveren. Vi har sat det op således:

²⁰ <http://docs.identityserver.io/en/release/>

```
// Role based Authorization: policy based role checks.
services.AddAuthorization(options =>
{
    // Policy for dashboard: only administrator role.
    options.AddPolicy("Manage Accounts", policy => policy.RequireRole("administrator"));
    // Policy for resources: user or administrator roles.
    options.AddPolicy("Access Resources", policy => policy.RequireRole("administrator", "user"));
});
```

Design angular komponent til login

I udviklingen af et komponent til login, fandt vi et kodebibliotek “angular-oauth2-oidc”²¹, dette kodebibliotek hjælper os med at implementere login feature til vores angular frontend app. Login kalder på en signin.ts klasse, som benytter et password authentication flow fra kode biblioteket, dette igangsætter på hele login processen, ved hjælp af dette kodebibliotek.

Support Web Desk

Login

admin@gmail.com

.....

☐ Remember me

Login

Angular komponent for bruger forside

Her har vi fulgt vores mockup, hvor en bruger bliver navigeret til denne efter login. På denne side kan brugeren se antal af nye tickets, samt hvor mange tickets der er tildelt denne bruger. De tickets som er tildelt denne bruger vil så se den i en datatabel på denne forside.

Support Web Desk

Startside

Tickets

Ticket

Logout

Startside

Velkommen admin@gmail.com

Nye Tickets

5

Dine Tickets

6

Filter

Status	Prioritet	Overskrift	Anmoder	Agent	Sidst opdateret	Se Ticket
Igang	Normal	7FrFnz	admin@gmail.co m	admin@gmail.co m	16:13 15-11- 2017	Se Ticket
Åben	Normal	mYPwIT	admin@gmail.co m	admin@gmail.co m	16:13 15-11- 2017	Se Ticket

²¹ <https://github.com/manfredsteyer/angular-oauth2-oidc>

Oprette et API endpoint til at trække tickets ud til bestemt bruger

Hertil har vi oprettet et par endpoints i backenden på vores ticket controller, hvor man kan udtrække data ved brug af et brugernavn:

```
/// <summary>
/// GET: api/Tickets/assigneeamount/{username}
/// </summary>
/// <returns>Returns amount of tickets that are assigned to the "user"</returns>
[HttpGet("assigneeamount/{username}")]
0 references | Bang0123, 5 days ago | 1 author, 1 change | 4 requests | 0 exceptions
public async Task<ActionResult> GetAssigneeTicketAmount([FromRoute] string username)...

/// <summary>
/// GET: api/Tickets/assignee/{username}
/// </summary>
/// <returns>Returns all tickets, sorted by updatedat dsc</returns>
[HttpGet("assignee/{username}")]
0 references | Bang0123, 5 days ago | 1 author, 1 change | 4 requests | 0 exceptions
public IList<TicketViewModel> GetAssigneesTickets([FromRoute] string username)...
```

Ovennævnte brugernavn kommer fra AuthenticationService i Angular appen, hvor den også henter AuthorizationHeaderen fra, som den bruger over således:

```
getAssigneesTickets() {
  const apiUrl = '/api/tickets/assignee/';
  return this.http.get(
    this.baseUrl + apiUrl + this.authService.getUser().userName, {
      headers: this.authService.getAuthorizationHeader()
    });
}

getAssigneesTicketsAmount() {
  const apiUrl = '/api/tickets/assigneeamount/';
  return this.http.get(
    this.baseUrl + apiUrl + this.authService.getUser().userName, {
      headers: this.authService.getAuthorizationHeader()
    });
}
```

Denne AuthenticationService er vores frontends authorization handler og data service, den er implementeret med kodebibliotek “angular-oauth2-oidc” i tanke, så denne kan servicere andre klasser med brugerinformation.

Testing

I denne user story benytter vi nogle kode biblioteker og et identity framework, som vi skynder ikke giver mening at unit eller integration teste. Derimod skal vi blot unit teste 2 angular componenter, derefter opstille et par acceptance tests, som skal implementeres i en senere sprint.

Unit test

I disse unit tests undersøges der om manipulation med “login” og “home” siderne er mulig. Her ses resultatet af de unit tests for begge componenter:

11 specs, 0 failures

```
HomeComponent
  should create the component
  should render h1 'startside'
  should render greeting text 'Velkommen Tester'
  should render text for new tickets and display amount
  should render text for assigned user tickets and display amount
  should render text for loading
  should render one row in datatable

LoginPageComponent
  should create the component
  should render h2 for login form
  should render inputs for login form
  should render login button for login form
```

Acceptance test

Vi har opstillet følgende tests:

- Login view should display Login screen when navigated to the site
- Login view should Login and see the redirected page
- Login view should logoff
- Home view should see greeting me `Admin`
- Home view should see new tickets
- Home view should see my tickets
- Home view should see my tickets in datatable

Sprint 3

Sprint planning

I denne sprint planlægger vi at gøre det en smule anderledes. I sprint 3 er det planlagt at teamet skal i gang med user story 7. Denne user story er estimeret til at være en af de store i projektet på 13 story points. Da vi grundet størrelsen på den pågældende user story ikke når at færdiggøre noget i denne sprint, vælger vi at tilsidesætte nogle af de normale scrum "events" såsom sprint review og retrospective. Dette gør vi for at være ekstra effektive i vores produktion af den planlagte user story.

Teamet havde diskuteret om hvorvidt der skulle laves planlægning i starten af næste sprint, eller om der skulle programmeres og holdes scrum møder som det eneste. Der blev enighed om at mandagen i næste uges sprint, altså sprint 4, var en god mulighed for at lave en opsummering på det opnåede arbejde og planlægge vores tid mere nøjagtigt, når vi var færdige med sprint 3.

Product owner er blevet inkluderet i planlægningen og er enig med os i, at dette er en god måde at gøre det på, da et sprint review i sprint 3 ville betyde test af det samme produkt som i sprint 2 review uden de store ændringer.

vi lavede en meget grov planlægning af ugen da vi også løbende efter hvert scrum møde vil forsøge at opdatere vores sprint backlog. Mere om dette længere nede ved vores simplificeret version af denne.

Ugeplan Sprint 3	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
Opgaver:	Scrum møde Sprint planning Organisere sprint backlog. Tasks oversigt og tildeling	Påbegynde udvikling af user story 7's tasks.	Scrum møde Arbejde med Tasks	Arbejde med Tasks	Scrum møde Arbejde med Tasks

Sprint Backlog

Da vi i denne sprint begynder en stor user story vil der også være mange tasks. Nogle af disse tasks har vi svært ved at forudsige hvordan de vil blive udarbejdet da vi ikke har så stor erfaring med teknologien. Derfor har vi forsøgt at dele user storien op i relativt store tasks, og vil så løbende efter hvert scrum møde uddrage nogle mindre og mere håndterbare tasks, som kan løses til næste scrum møde. På den måde går vi heller ikke i gang med for stor en opgave af gangen og har overblikket over hvor langt vi er, med de forskellige tasks hele tiden. Den første version af denne sprints backlog ser således ud og vil blive ændret løbende på trello for at gøre det lettere for teamet at arbejde med.

User stories	Tasks	In progress	Done
US 7: Som supporter vil jeg kunne svare tilbage på supportsager inde i den pågældende ticket så jeg ikke lader kunder vente i uvished.	Kodes en mail service til at håndtere afsendelse af mails. Implementer mail formalia system. Baggrundsjob der tjekker mails og opretter dem som tickets eller beskeder tilhørende tickets. Mail tekst og dertilhørende information gemmes i ticket historik. Beskeder skal vises sammen med én ticket.		

	Besked Endpoint på API'en		
--	---------------------------	--	--

Scrum møder

Opsummering af Scrum møder

Mandag

Vi startede scrum mødet med at snakke omkring hvordan planlægningen skulle laves. Da vi begge har manglende erfaring med afsendelse og modtagelse af mails ved hjælp af C# vidste vi på forhånd at, dette kunne være noget der skulle arbejdes lidt ekstra med. Der var nok at tage fat i med hensyn til at få oprettet tasks ud fra den pågældende user story, og vi gik derfor hurtigt videre med dagens opgaver.

Onsdag

Midtvejs i sprinten holdte vi som planlagt ugens andet scrum møde. Vi briefede hurtigt hinanden omkring hvad der var blevet arbejdet med dagen før, og besluttede os for at gå videre i sprinten med pair programming. De næste opgaver krævede research og læren om nye måder at gøre tingene på, så vi tænkte at det var bedre med fire øjne i stedet for to.

Fredag

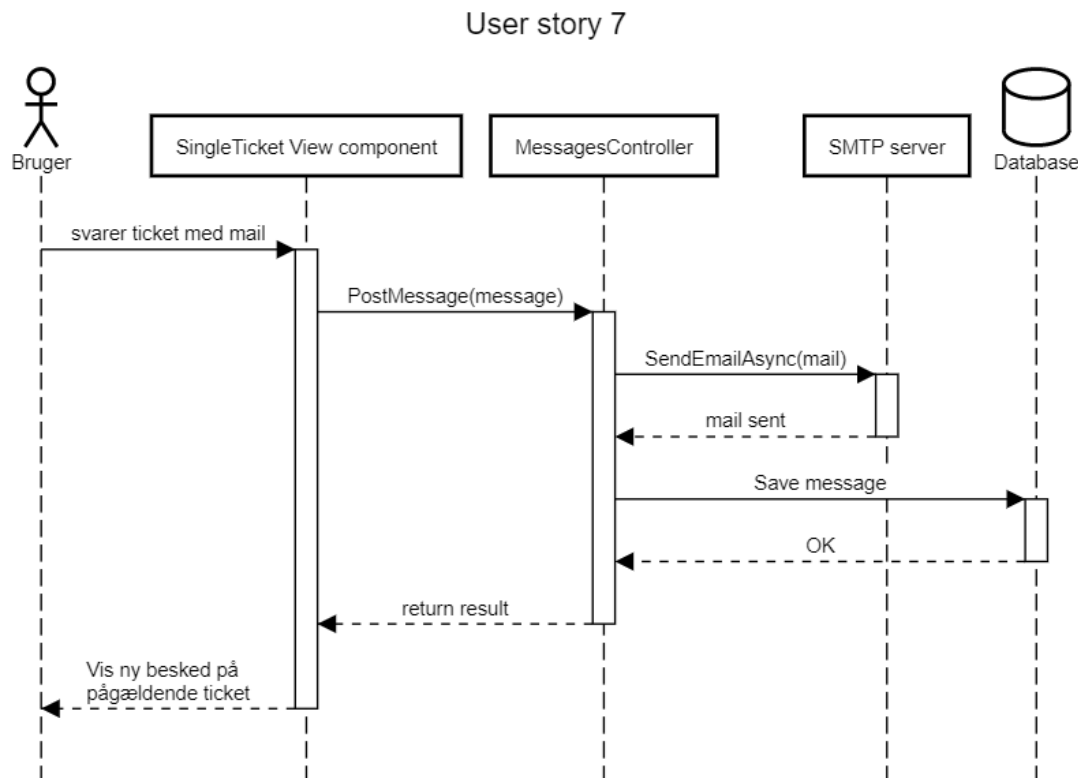
Grundet pair programming arbejdede vi sammen hele torsdagen og derfor blev det kun kort opsummeret hvad vi nåede sammen. Vi har fulgt vores planlægning indtil nu, men ser frem til planlægningen for næste sprint da det kan være med til at give os et bedre overblik end vi har på nuværende tidspunkt.

Fredagen var et anderledes scrum møde i forhold til tidligere fredags scrum møder. Normalt ville vi eksempelvis planlægge hvordan og hvornår sprint reviews skulle forløbe og ligeledes med retrospective. Men disse punkter har vi, som tidligere skrevet, valgt at tilsidesætte til fordel for programmering af produktet. Vi fortsatte derfor ud fra sprint planlægningen med flere tasks.

Tekniske detaljer

User story 7 består af seks tasks, disse tasks reflekterer PO's acceptance kriterier fint. Navnene på et par Tasks har været sætninger, så i denne tekniske gennemgang, er de blevet omformuleret og lagt under en mere generel titel. Derudover bliver et par af disse

først beskrevet i næste tekniske detaljer, da de ikke blev nået i denne. User story 7 beskriver et forløb, hvor det er muligt at svare på en besked fra en mail server, det er derfor relevant i denne teknisk gennemgang at få beskrevet den måde hvorpå vi gør dette med baggrund jobs, samt belyse visuelt hvordan user story 7 fungerer:



Baggrundsjobs

For ASP.NET Core er der ikke nogen implementeret routine, der kan køre baggrunds arbejde, så dertil har vi valgt at benytte os af Hangfire. Hangfire²² er et open-source backgroundworker framework, som nemt kan integreres og køres ved siden af en ASP.NET Core web applikation. Hangfire tilbyder nem adgang til at kunne udføre funktioner, som man ikke vil have på sin request pipeline. Vi benytter hangfire til at udføre funktioner tilpasset efter hvor ofte de ønskes kørt:

```
app.UseHangfireServer();
RecurringJob.AddOrUpdate(
    methodCall: () => new EmailServiceJob(ctx).Invoke(true),
    cronExpression: Cron.MinuteInterval(5)
);
RecurringJob.AddOrUpdate(
    methodCall: () => new TicketServiceJob(ctx, emailer).Invoke(),
    cronExpression: Cron.MinuteInterval(5)
);
```

²² <https://www.hangfire.io/overview.html>

Her i startup klassen, definerer vi at benytte hangfire, samt her tilføjer vi de to jobs, der skal køre hvert femte minut. For at manipulere med mail protokollerne imap og smtp, har vi benyttet MimeKit/MailKit²³, dette Mime “multipurpose internet mail extension” kodebibliotek har mange brugbare extensions og forbedringer af .NET’s eget kodebibliotek, derudover har det også virket perfekt og været nemt til formålet.

Mails gemmes og processeres

EmailServiceJob håndterer at hente mails fra den konfigurerede mailbox, samt markere dem “læst”, så jobbet ikke henter den samme mail mere end en gang. Hvorefter den oversætter mailene til Mail entities og gemmer disse entities i databasen.

TicketServiceJob sørger for at processere de mails der er gemt i databasen, den bestemmer om der skal oprettes en ticket, eller om den pågældende mail er et svar på en ticket. Dette gør den ved at bruge Regex, den kigger på subject i mailen, og finder ud af om der er et ticket id, og i så fald opretter den mailen som en besked til den ticket der er refereret i mail subject:

```
var match = Regex.Match(
    input: mail.Subject,
    pattern: @"T: \d+ M: \d+ \|");
if (match.Success)
{
    await CreateMessage(mail, match, _ctx);
}
else
{
    await CreateTicket(mail, _ctx);
}
```

/ T: \d+ M: \d+ \| / g
T: matches the characters T: literally (case sensitive)
 ▶ \d+ matches a digit (equal to [0-9])
 + Quantifier — Matches between one and unlimited times, as many times as possible, giving back as needed (greedy)
M: matches the characters M: literally (case sensitive)
 ▶ \d+ matches a digit (equal to [0-9])
 matches the character \| literally (case sensitive)
 \| matches the character \| literally (case sensitive)

Hvis der ikke er noget match, må denne ticket bare være ny og skal behandles som en ny ticket, hvor der oprettes sådan en i systemet og sendes et autosvar til anmoder.

Alt dette sker helt uden at forstyrre vores web applikations request pipeline og dens performance, hvilket er vigtigt i og med vi benytter regex, regex kan være utrolig CPU tung, så derfor skal man helst bruge det med omtanke. Den omtanke har vi udvist i og med vi forsøger at returnere efter første match, dvs at hvis et subject følger vores pattern, vil der ikke blive brugt meget tid på at søge, og vi kan dermed hurtigt bestemme typen af mail.

Mail service til afsendelse af mails og formalia

Her har vi kodet en service der kan sende emails, denne er decoupled og kan derved leveres igennem asp.net core’s dependency injection.

²³ <http://www.mimekit.net>

```
// Email registered for dependency injection
services.AddTransient<EmailSender, EmailSender>();
public interface IEmailSender
{
    2 references | Bang0123, 7 days ago | 1 author, 1 change | 0 exceptions
    Task AutoReply(string email, string requester, int ticketId, string subject);
    3 references | Bang0123, 5 days ago | 1 author, 1 change | 0 exceptions
    Task<bool> SendEmailAsync(MimeMessage message);
    3 references | Bang0123, 5 days ago | 1 author, 1 change | 0 exceptions
    string GetFormattedSubject(int ticketId, int msgId, string subject);
    2 references | Bang0123, 5 days ago | 1 author, 1 change | 0 exceptions
    string AttachSignature(string body, string signature);
    2 references | Bang0123, 5 days ago | 1 author, 1 change | 0 exceptions
    string Formatbody(string body);
}
```

Denne EmailSender kan ved brug af MimeKit/MailKit kode biblioteket, sende mails igennem vores valgte smtp server, i dette tilfælde Gmails smtp server.

For at implementere et formalia system, er der blevet tilføjet en property til en bruger, som indeholder deres personlige signatur. Denne signatur bliver sat sammen med den formaterede body, så denne bliver til mailens indhold.

Herunder ses eksempel på brug:

```
var txtbody = _emailer.AttachSignature(
    body: _emailer.Formatbody(body: message.Body),
    signature: ticket.Assignee.EmailSignature
);
builder.TextBody = txtbody;
mime.Body = builder.ToMessageBody();
```

Testing

Denne user story har været ret svær at teste, så vi har kun opstillet en acceptance test, fordi det gav mening, hvorimod unit tests ikke helt giver mening, da vi skal interagere med tredjeparts systemer. Der kunne til fordel opstillet et par integration tests, med mocked data, men dette har vist sig ikke at være lige til.

Acceptance test

Følgende test er blevet opstillet:

- Single ticket view should send extern message and see it afterwards

Sprint 4

Sprint planning

Sprint planlægningen for denne sprint blev brugt som et opsummerings møde for sidste sprint og videre planlægning af tasks for user story 2. Vi kunne se at arbejdet i sprint 3 ikke helt fulgte med tidsplanen og derfor blev vi nødsaget til at planlægge med nogle længere arbejdsdage for at nå det "tabte" arbejde.

Derudover så vi frem til at afholde sprint review og retrospective, da vi kan mærke at vi lidt manglede det som afrunding på sidste sprint. Som noget af det sidste i sprint 4, vil vi som i sprint 2, også afholde brugertest som er beskrevet tidligere.

Ugeplan Sprint 4	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
Opaver:	Scrum møde & Opsummering Sprint planning Organisere sprint backlog. Tasks oversigt og tildeling	videre arbejde med tasks fra sprint 3, tilhørende User story 7.	Scrum møde videre arbejde med tasks fra sprint 3, tilhørende User story 7.	<u>Færdiggørelse</u> af tasks fra sprint 3, tilhørende User story 7.	Scrum møde påbegynde arbejde med Tasks for user story 2. Sprint Review Brugertest Retrospective

Sprint Backlog

I tabellen nedenfor kan man se sprint backloggen som den så ud da sprint 4 satte i gang. I rækken for user story 7 kan man se at, der ikke var nået så langt med de forskellige tasks som der var ønsket. Derudover er user story 2 blevet opdelt til mindre tasks og tilføjet backloggen da produktionen af denne user story starter i denne sprint.

User stories	Tasks	In progress	Done
US 7: Som supporter vil jeg kunne svare tilbage på		Beskeder skal vises sammen med én ticket.	Kodes en mail service til at håndtere afsendelse af

supportsager inde i den pågældende ticket så jeg ikke lader kunder vente i uvished.		Besked Endpoint på API'en Mail tekst og dertilhørende information gemmes i ticket historik.	mails. Baggrundsjob der tjekker mails og opretter dem som tickets eller beskeder tilhørende tickets.
US 2: Som supporter vil jeg kunne skrive noter til tickets løbende så jeg har overblik over alle sagens tiltag.	Gemte noter skal kunne vises på en ticket. Noter API Endpoints. implementer note funktionalitet i eksisterende angular service. implementer backend noter model.		

Opsummering af scrum møder

Mandag

På dette scrum møde lagde scrum master vægt på at det var vigtigt at teamet hver i sær fik svaret på alle tre spørgsmål ud fra scrum håndbogen. Da teamet følte de var bagud mente han at det var vigtigt at prioritere kommunikation højt, så der var helt styr på hvad der blev lavet og hvor vi skulle hen med arbejdet. Vi startede med at briefe hinanden omkring arbejdet vi afsluttede sidste sprint med. De to tasks kan ses i sprint backloggen under kolonnen "done". Til sidst blev vi enige om at afslutte scrum mødet og få lavet sprint planlægning så vi kunne komme videre med arbejdet.

Onsdag

Grundet god løbende kommunikation de to første dage af sprinten blev onsdagens scrum møde meget kort. Teamet havde et godt og tæt samarbejde der gjorde at alle parter var med i hvad der var lavet. Derfor brugte teamet kun lidt tid på at finde ud af om der var nogle udfordringer vi skulle være opmærksomme på og gik så videre med dagens arbejde som det var planlagt i sprint planlægningen.

Fredag

Grundet pair programming med de sidste opgaver til user story 7, vidste teamet godt hvad der var blevet lavet de to forgangne dage. Vi opsummerede dog kort sammen hvad vi havde

fået lavet for at være sikre på vi ikke have glemt noget. Acceptance kriterierne var mødt og vi var klar til at se fremad med user story 2. Til sidst snakkede vi om hvordan vi skulle få afviklet sprint review og bruger test så vi kunne få noget tiltrængt feedback på arbejdet.

Sprint Review

Det længe ventede sprint review blev startet med at hoppe direkte til product owners test af det udførte arbejde. Han gik hurtigt over det arbejde, som han havde givet feedback på i tidligere sprints og kunne se at teamet havde fulgt op på de ændringer, som han havde ønsket. Herefter kiggede han på de to user stories, som vi havde implementeret i sprint 3 og 4. Product owner testede hvordan det var at sende mails fra programmet og kunne se at den afsendte mail også blev tilføjet til den pågældende tickets historik. Han var overordnet tilfreds med funktionaliteten af user story 7, men manglede lidt genkendeligheden i designet fra de tidligere udarbejdede mock ups²⁴. Som han pointerede for teamet: "Alt funktionaliteten er der, men måden hvorpå user interfacet er arrangeret stemmer ikke helt overens med prototypen?". Efter en god snak omkring design og funktionalitet blev vi enige om at design og styling af ticket viewet ville blive taget op igen i en senere sprint. Dette gøres for at gøre det lettere at overholde tidsplanen for de kommende user stories. Derudover giver det også mere frirum i user interfacet, hvis der skal tilføjes mere funktionalitet senere i produktets udvikling.

Da product owner allerede befandt sig i ticket viewet testede han naturligvis også muligheden for at tilføje noter til en ticket, altså user story 2. Da funktionen minder utrolig meget om måden hvorpå man sender mails, fandt han hurtigt ud af det og det virkede efter hensigten. Efter test af de to user stories gennemgik han de forskellige acceptance kriterier for begge stories og kunne se at alle kriterierne var opfyldt.

Brugertest

Efter vores sprint review er det blevet tid til endnu en brugertest²⁵. På samme måde som sidst blev vores testperson stillet nogle opgaver som skulle løses. Da opgaverne er relativt korte og tidligere har vist sig at være hurtige at løse, vil vi i denne test også stille de opgaver som blev lavet til de foregående user stories. Dette gjorde vi da det kunne være den nye testperson oplevede navigeringen af disse anderledes end den forrige testperson. på den

²⁴ Se bilag 1

²⁵ Detaljerne for hvordan brugertesten bliver afholdt er beskrevet i sprint 2.

måde får vi også mere ud af vores bruger tests, når der er sat tid af til dem i vores planlægning.

Som testen skred frem kunne vi se at testpersonen var effektiv i brugen af programmet og hurtigt fandt ud af hvordan det skulle bruges. Som sidste test person var han hurtigt færdig med opgaverne og vi indledte derfor til en snak omkring hans oplevelse af produktet.

Testpersonen blev bedt om at komme med feedback på produktet steder, hvor han kunne se at der var plads til forbedringer. Der blev givet udtryk for at produktet i sit nuværende stadie og med de opgaver han blev stillet var nemt at have med at gøre. Det var let at overskue og i opbygningen af programmet kunne han godt lide at alt navigering fandt sted i navigationsmenuen til venstre på skærbilledet.

Testpersonen kunne dog også se nogle mangler ved programmet. Den ene ting han påpegede var at man eksempelvis ikke kunne ændre hvem der var ansvarlig for en ticket. Som systemet ser ud nu er det den supporter der senest har handlet på en ticket som vil blive tildelt rollen som ansvarlig. Dette er sikkert ikke noget product owner har tænkt over før og er derfor aldrig blevet tilføjet til product backloggen som en user story. Teamet er nu opmærksomme på at dette kunne være en god mulighed og kan nu gå videre med ideen til product owner.

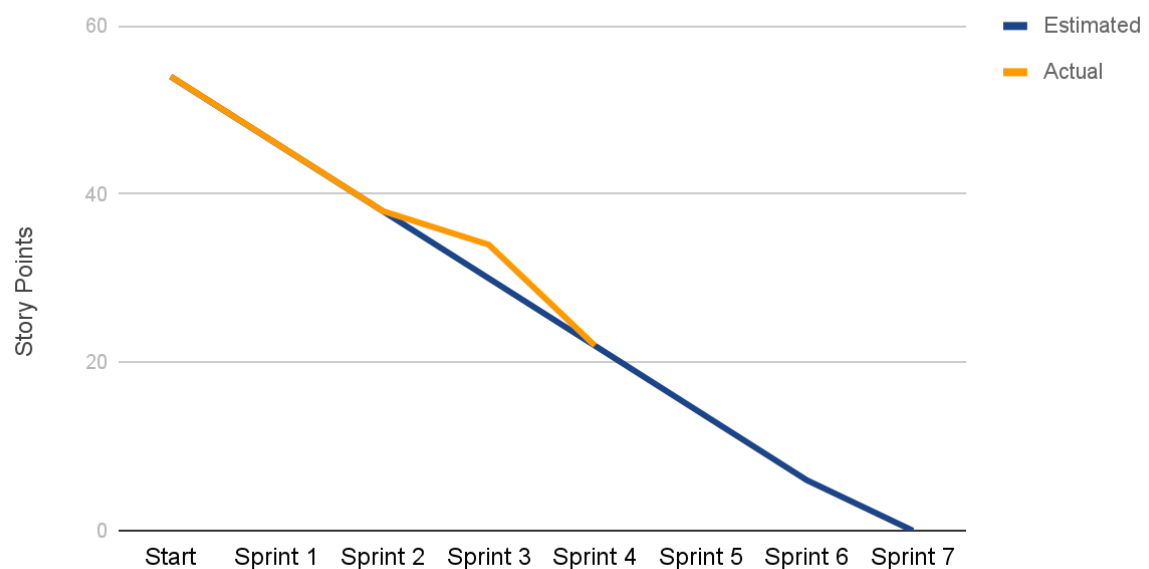
Alt i alt har brugertesten endnu en gang været en god ting at få afviklet, da vi har fået feedback og kritik af det som er blevet udviklet. Det er rart at få et ekstra sæt øjne på produktet da det kan være med til at belyse nogle problemstillinger som teamet kan have overset.

Burndown Chart

Teamet har rigeligt at lave, dog har ingen deadline i sprint 3 reflekteret, at mere af arbejdet har trukket ud. Vi nåede i sprint 3 kun 4 ud af 8 story points, hvilket ikke var optimalt. Dette fik vi indhentet i sprint 4, hvor vi nåede i alt 12 story points, hvor 4 var fra sprint 3. Teamet har en god mulighed for at opretholde den gode pace, dette forudsætter produktivitet og engagement.

Burndown Chart

End of Sprint 2



Sprint Retrospective

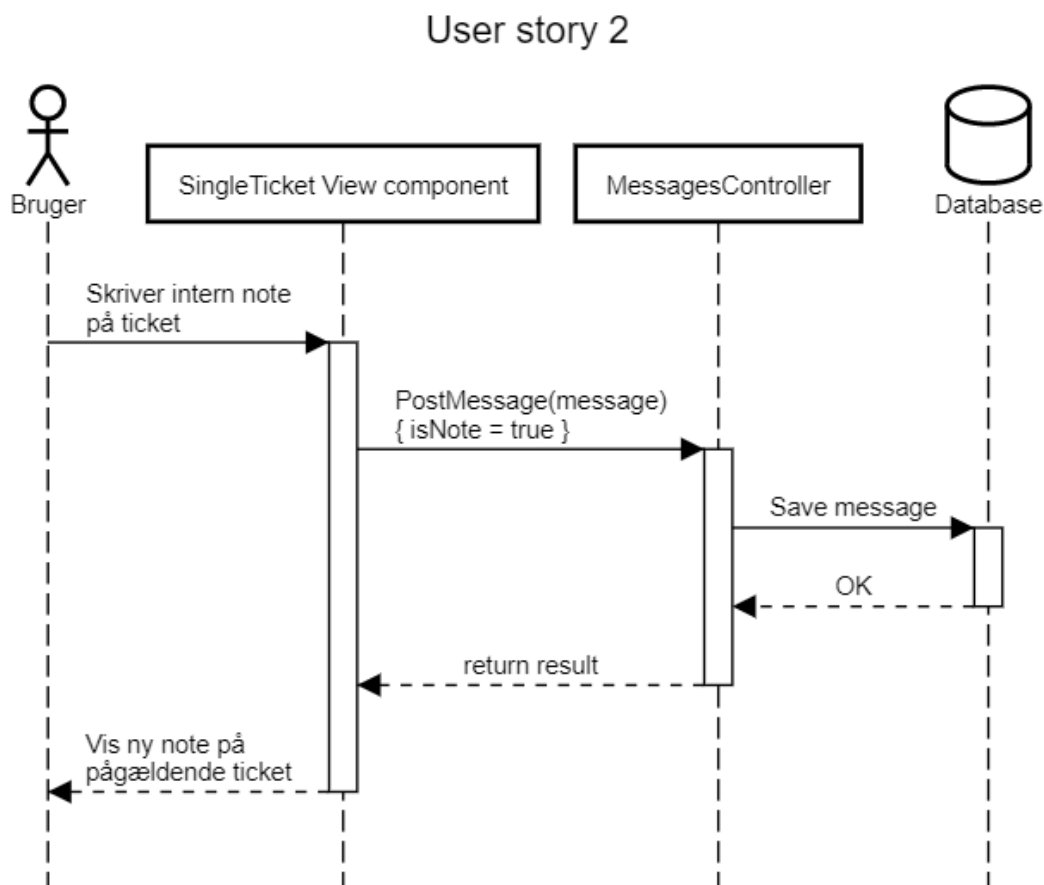
Som scrum master indledte Nikolaj med at opsummere hvordan sprint 3 og 4 var blevet afviklet. Der var to primære emner som han påpegede; Teamets arbejde som gruppe og teamets velocity. Observeringen på teamet som gruppe var god. Der er fortsat et godt samarbejde og kommunikationen virker efter hensigten. Dette kan også skyldes at vi, specielt i sprint 4, havde fokus på kommunikation da teamet havde nogle story points at skulle indhente.

Det andet punkt som han påpegede var der dog en del snak frem og tilbage omkring. Som team kunne vi se at vores velocity i sprint 3 ikke havde været god nok. Det planlagte arbejde blev simpelthen ikke nået og det var ikke på grund af dårlig planlægning. Hvis teamet så indad var der enighed om at vi hver især ikke havde arbejdet så effektivt som vi kunne. Vi diskuterede længe årsagen til at teamet ikke havde været lige så "motiveret" i netop sprint 3 som i de andre sprints. Der var enighed om at en af faktorerne kunne være at der ikke var noget sprint review og fremvisning af arbejdet til sidst i sprinten. Da der pludselig var to uger imellem sprint reviews kan teamet indirekte have følt at de havde god tid og det gjorde at man kom til at sætte velociteten ned. Det resulterede i at vi ikke var effektive nok og derfor fik noget arbejde at indhente i sprint 4.

Vi vil forsøge at mitigere dette problem i fremtidige sprints ved kontinuerligt at afholde sprint reviews hver fredag og ikke nedprioritere et sådan møde og undervurdere effekten af det.

Tekniske detaljer

Her beskrives de sidste tasks fra user story 7, samt tasks fra user story 2. Det drejer sig i alt om seks tasks, de bliver beskrevet en mere generelle overskrifter. Disse Tasks har kun reflekteret PO's acceptance kriterier nogenlunde, fordi der ikke blev taget hensyn til data modelleringen fra besked og noter.

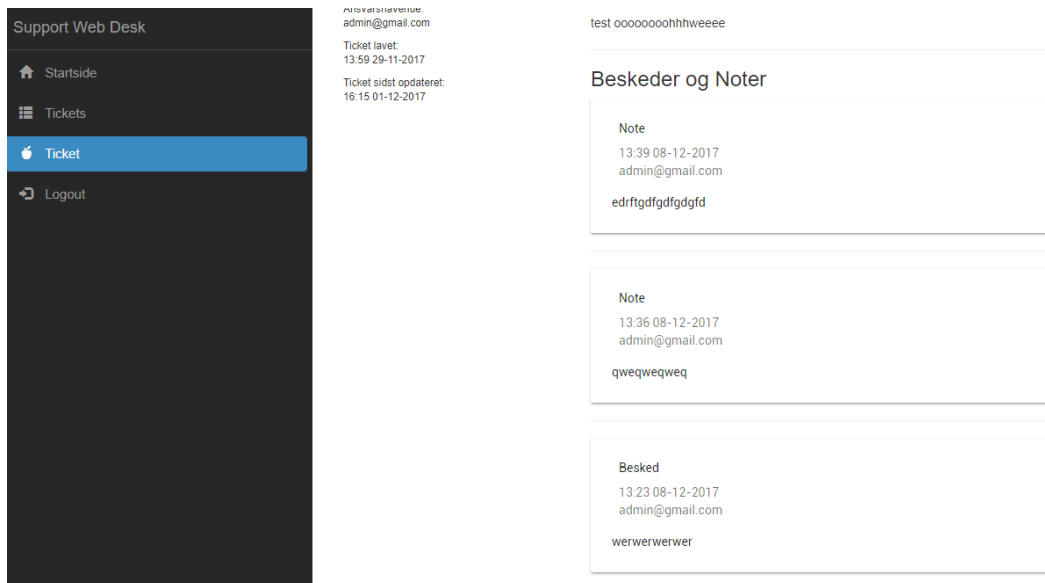


Beskeder og noter implementeringen

Efter en revidering af redundant arbejde fra beskeder til noter, har vi valgt at disse 2 modeller ligner hinanden så meget, at vi har valgt at lægge dem sammen til en "message" model og udstyre den med en property, der beskriver om den er en note eller en besked. Her har vi oprettet et API endpoint, dette endpoint returnerer alle messages for en ticket, når man giver den et ticket id.

```
/// <summary>
/// GET: api/Tickets/messages/{id}
/// </summary>
/// <returns>Returns all ticket messages for an id, sorted by updatedat dsc</returns>
[HttpGet("messages/{id}")]
0 references | Bang0123, 5 days ago | 1 author, 1 change | 0 requests | 0 exceptions
public async Task<IActionResult> GetTicketMessages([FromRoute] int id)...
```

Hvorefter vi har implementeret at hente fra denne i vores ticket service, denne service bliver allerede benyttet i vores singleticketview, så det var oplagt at ligge den her.



Der eksisterer logic i angular html template, som bestemmer om der er tale om en note eller en besked, dette kan derved give en besked og note hvert sit udseende.

```
<mat-card-title *ngIf="message.isNote; then notetemp; else beskedtemp"></mat-card-title>
<ng-template #notetemp>
  <mat-card-title>Note</mat-card-title>
  <mat-card-subtitle>{{message.updatedAt | date:'HH:mm dd-MM-yyyy'}}
  <br> {{message.author}}
</mat-card-subtitle>
</ng-template>
<ng-template #beskedtemp>
  <mat-card-title>Besked</mat-card-title>
  <mat-card-subtitle>{{message.updatedAt | date:'HH:mm dd-MM-yyyy'}}
  <br> {{message.sender}}
  <br> {{message.senderEmail}}
</mat-card-subtitle>
</ng-template>
```

Testing

Ligesom den forrige har denne user story ikke været til at unit teste, dog ville det have været oplagt at opstille nogle integration tests, da vi tester et forløb med backenden. Integration tests har vist sig ikke at være lige til, da vi har problemer med at få mocked og faked data igennem falske providers i angular. Derudover har vi opstillet en acceptance test, da vi senere vil teste systemet med disse helheds tests.

Acceptance test

Følgende test er blevet opstillet:

- single ticket view should send intern message and see it afterwards

Sprint 5

Sprint planning

Som i tidligere sprint startede vi sprint 5 med at planlægge vores arbejde. I denne sprint er det user story 6 der vil blive arbejdet på og den er estimeret til 8 story points hvilket passer godt med teamets velocity. Det vil sige at user story 6 vil blive produceret og har planlagt release i denne sprint. Det er tydeligt at teamet har udarbejdet sprint planlægningen sammen flere gange. Det er en process der hurtigt bliver færdig og det er også fordi at teamet nærmest har lavet en fast "skabelon" som de arbejder ud fra. Forskellen fra tidligere sprint og denne er blot hvilken user story der vil blive arbejdet på.

Ugeplan Sprint 5	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
Opaver:	Scrum møde Sprint planning Organisere sprint backlog. Tasks oversigt og tildeling	Påbegynde udvikling af user story 6's tasks.	Scrum møde Arbejde med Tasks	Arbejde med Tasks	Scrum møde Arbejde med Tasks Sprint Review + Retrospective

Sprint Backlog

Efter den planlagte sprint planning gik teamet i gang med sprint backloggen. User story 6 skulle opdeles i tasks så opgaven blev mere overskuelig og lettere at uddelegere til udviklerne. Herunder kan man se sprint backloggen, som den så ud i starten af sprinten umiddelbart efter udarbejdelsen.

User stories	Tasks	In progress	Done
US 6: Som supporter vil jeg kunne søge efter tickets så jeg hurtigt kan finde en support sag.	Design search query parametre implementer backend endpoints Design Angular component for search implementer Angular søge service		

Opsummering af scrum møder

Mandag

Grundet en god dialog i afslutning af sprint 4 valgte teamet at det ikke var nødvendigt at briefe hinanden om det allerede udførte arbejde fra sidste sprint. Derimod blev teamet enige om at gå direkte i gang med dagens arbejde og senere tilskrive en task til hver udvikler som de kunne arbejde med dagen efter.

Onsdag

Onsdagens scrum møde blev holdt lige efter scrum håndbogens ide omkring hvad et scrum møde er. Teamets medlemmer startede med at gennemgå hvad de havde opnået dagen før. Andreas briefede omkring hvad han havde udarbejdet i user interfacet for tasken "*Design Angular component for search*" og Nikolaj viste ganske kort den kode han havde fået skrevet til hans egen task, "*implementer Angular søge service*" om tirsdagen. Herefter blev der aftalt hvordan resten af dagens arbejde skulle udføres og om der var nogle udfordringer som teamet skulle være opmærksomme på.

Fredag

Fredagens scrummøde blev åbnet med en hurtig opsummering på hvad der var lavet onsdag og torsdag. Teamet er godt med i forhold til sprintens planlagte arbejde og blev enige om, at det sidste kode og tests ville blive gjort ved hjælp af pair programming. Til sidst kunne Andreas informere om at product owner havde meldt ud at han ikke havde mulighed for at være en del af sprint reviewet denne gang. Teamet blev derfor enige om at gøre som i en tidligere sprint og lade Andreas sidde stand in for product owner.

Sprint review

Sprint reviewet gik i gang umiddelbart efter det færdige arbejde var blevet testet. Som tidligere nævnt sidder Andreas stand in for product owner, så der var ikke nogen ude fra som teamet skulle vente på for at komme i gang.

Sprint reviewet startede med en gennemgang af det gennemførte arbejde. Dette blev gjort ved at lade product owner prøve applikationen for at se om han kunne lide resultatet. Umiddelbart efter at have prøvet produktet gennemgik han de forskellige acceptance kriterier for user storiene og kunne se at der var taget højde for det hele. Som det sidste kiggede han på de fælles udarbejdede mockups fra sprint 0 og kunne se at user interfacet lignede det aftalte design. Han kunne derfor, med alle ovennævnte kriterier tjekket, godkende user storiens status som værende "done".

Teamet ville gerne have planlagt fremtidige sprint med den rigtige product owner da der igennem det udførte arbejde indirekte er blevet færdiggjort en user story.

Det drejer sig om user story 5: *"Som supporter vil jeg kunne se en tickets historik så jeg har overblik over hvilke tiltag der tidligere er blevet gjort."*

Formålet med at se en tickets historik bliver dækket i to forskellige tidligere udarbejdede user stories. Senest er det user story 6 for denne sprint. Efter at have søgt på en ticket får man mulighed for at vælge et resultat fra listen under søgekriterierne²⁶. Efter at have valgt den eftersøgte ticket vil applikationen navigere brugeren til "Ticket" viewet²⁷, hvor man kan se historikken for den pågældende ticket.

Næste sprint, altså sprint 6, indebar planlægning af user story 5 som nu allerede er udført. Derfor vælger teamet at forkorte projektet med en sprint og gå videre med det planlagte arbejde for sprint 7. På denne måde får product owner også sit product hurtigere klar til release. Efter endt sprint review kunne Andreas som product owner standin, kommunikere dette videre til den rigtige product owner og høre hvad han mente om ideen.

Burndown chart

I denne sprint blev der udført arbejde estimeret til 8 story points, teamets pace har været høj og god. Produktet er også begyndt at ligne noget, samt er der bygget et fundament, som gør det noget nemmere at tilføje nye features til. Stadiet i udviklingsprocessen forhøjer også

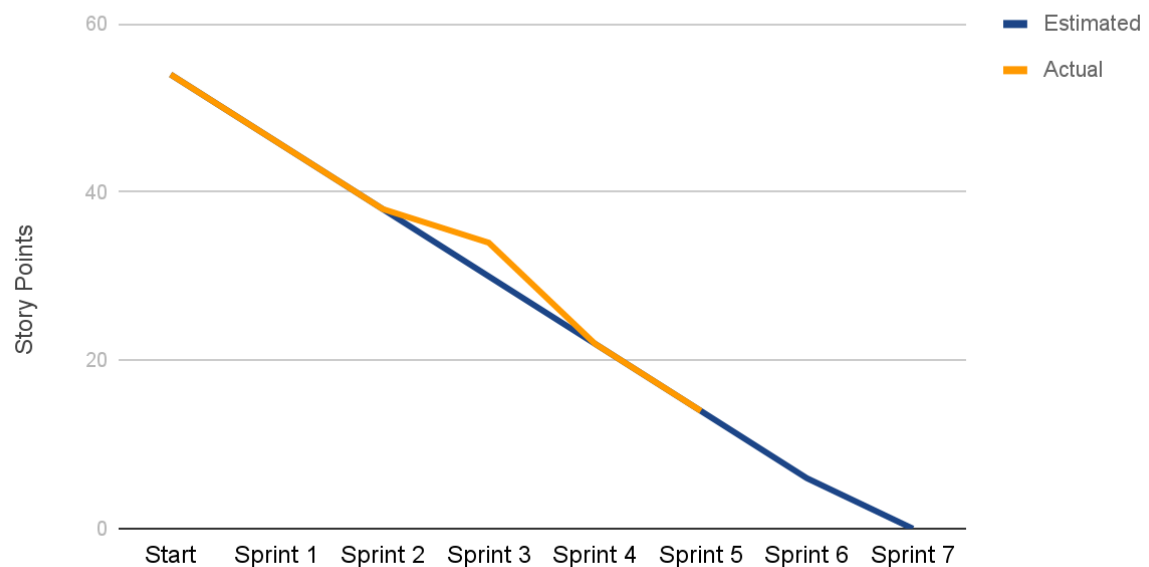
²⁶ Se bilag 5

²⁷ Se bilag 1.3

pacen, da teamet muligvis har stødt på lignende tasks tidligere i udviklingsprocessen, som de derved er bedre stillet til at løse.

Burndown Chart

End of Sprint 2



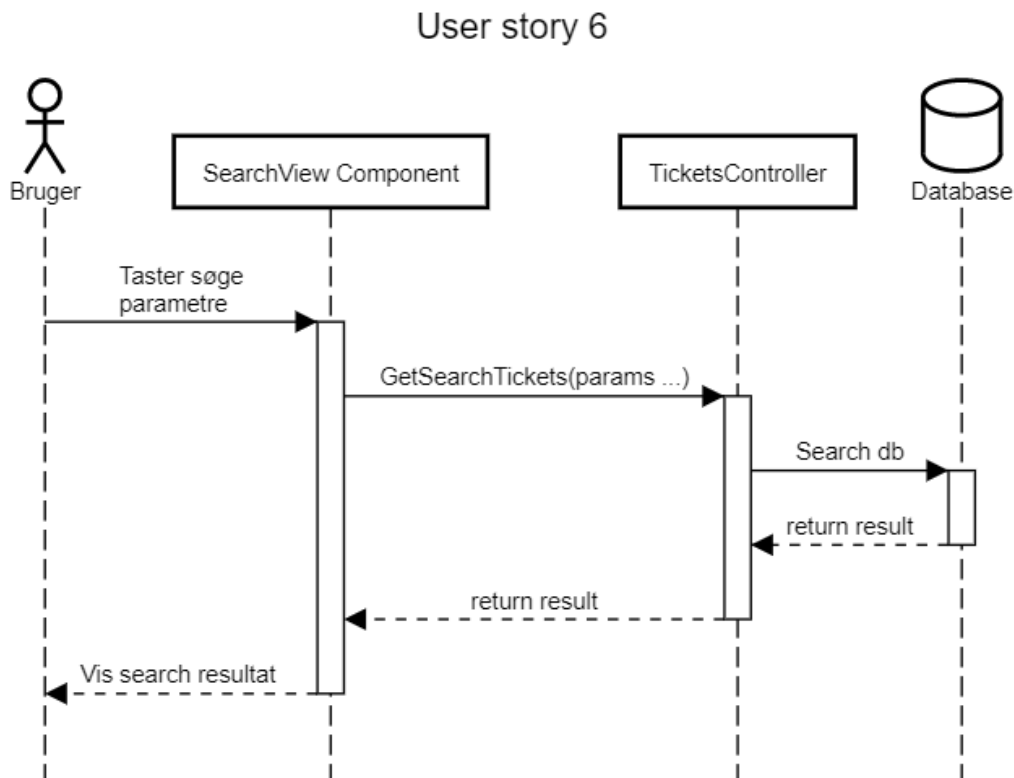
Sprint retrospective

Sprint 5 har været en super god oplevelse for teamet. Der har været godt individuelt arbejde men også en smule pair programming, hvor der har været sparret godt med hinanden i kodnings processen. Pair programming er en arbejdsform som vi brugte meget flittigt tidligere i de første sprints og er noget som teamet måske vil gå mere tilbage til da effektiviteten ved arbejdet er høj og kvaliteten af koden også stiger i takt med at der er flere øjne på arbejdet.

I en tidligere sprint havde teamet sænket sin velocity og havde pludsligt arbejde at skulle indhente i den efterfølgende sprint. Dette er en fejltagelse som der også har været fokus på og teamet har derfor arbejds-mæssigt været rigtig godt med i sprint 5. At være på forkant med arbejdet føler vi har givet en højere moral i teamet. Derudover har den fælles efterstræben på at være effektive, været utroligt motiverende for alle parter.

Tekniske detaljer

I denne tekniske detaljer vil beskrive udviklingen af user story 6, hvor der bliver implementeret et separat view til søgning i databasen. Denne user story er opdelt i 4 fire tasks, disse bliver beskrevet under overskrifter om den hører til frontend eller backend. Taskene har reflekteret PO's acceptance kriterier fint, hvilket har lagt op til en nem tilgang til hvad PO ønskede. Måden hvorpå user story 6 skal virke, kan skitsere ved følgende figur:



Implementer search component med søgefunktionalitet

Her har vi designet et component, med udgangspunkt i vores prototype med et søge view. Vi har mest benyttet en del input elementer fra Material angular, samt den gode datatabel til at vise resultaterne i.

Support Web Desk

Startside
Tickets
Ticket
Søg efter tickets
Logout

Søg efter tickets

Ticket id: 32 Overskrift: Prioritet: ▼ Dato fra: Søg

Anmoder: Fri text: Ansvarshave...: Dato til: Søg

Filter tickets

Status	Prioritet	Overskrift	Anmoder	Agent	Sidst opdateret	Se Ticket
Igang	Normal	Hej	Nikolaj Bang	admin@gmail.com	15:20 07-12-2017	Se Ticket

Items per page: 5 1 - 1 of 1 < >

Når man klikker på denne søgeknop, samler componentet de indtastede parametre sammen, som den sender videre til en service, der foretager et http kald mod et backend endpoint.

```
searchTickets(searchParams: HttpParams) {  
  const apiUrl = this.baseUrl + '/api/Tickets/search';  
  return this.http.get(  
    apiUrl, {  
      headers: this.authService.getAuthorizationHeader(),  
      params: searchParams  
    });  
}
```

Implementering af backend søge funktioner

For at implementere en dynamisk søge metode, der kun søgte på de parametre som frontenden sender, har vi benyttet kodebiblioteket “LINQKit”²⁸, som indeholder en masse udvidelser af de allerede eksisterende “linq to sql”, som findes i entity framework. Her benytter vi en udvidelse primært med fokus på at bygge et “predicate” til det “where” statement, som søger i databasen. Denne udvidelse hedder et såkaldt “PredicateBuilder”²⁹, som gør det muligt at lave dynamiske queries, alt efter logik og validation som en controller kan bestå af.

²⁸ <http://www.albahari.com/nutshell/linqkit.aspx>

²⁹ <http://www.albahari.com/nutshell/predicatebuilder.aspx>

```
/// <summary>
/// GET: api/Tickets/search
/// </summary>
/// <returns>Returns all tickets, sorted by updatedat dsc</returns>
[HttpGet("search")]
0 references | Bang0123, 13 days ago | 1 author, 1 change | 2 requests | 0 exceptions
public IList<TicketViewModel> GetSearchTickets(
    [FromQuery] int? ticketid,
    [FromQuery] string requester,
    [FromQuery] string subject,
    [FromQuery] string priority,
    [FromQuery] string body,
    [FromQuery] string assignee,
    [FromQuery] DateTime datefrom,
    [FromQuery] DateTime dateto)
{
    var pred = PredicateBuilder.New<Ticket>();
    if (ticketid != null)
    {
        pred = pred.Or(ticket => ticket.Id == ticketid);
    }
}
```

Ovenfor kan vi se metoden, som håndterer endpointet “api/search”, denne metode får en masse parametre fra et http kald’s “Query” parametre. ASP.NET frameworket håndterer at populære metodens parametre, hvis de findes i det indkommende http kald.

I starten af denne metode oprettes et nyt “PredicateBuilder” objekt, så vi derefter kun sætter de nødvendige “expressions” på, dette kan gøres med “And” og “Or” statements.

Hvorefter man til sidst giver dette “PredicateBuilder” objekt til den database query, man vil udføre.

```
var ticks = _context.Tickets
    .Include(ticket => ticket.Assignee)
    .Where(pred)
    .OrderByDescending(ticket => ticket.UpdatedAt)
    .Distinct();
```

Testing

Denne user story testes ved brug af unit tests og acceptance tests, unit tests over selve angular search componentet, samt opstilles der nogle acceptance tests til implementering i næste sprint.

Unit test

Disse unit tests inkludere angular componentet “search”, her testes der udelukkende på hvordan componentet kan manipuleres med data.

Acceptance test

5 specs, 0 failures

SearchViewComponent

- should create the component
- should render h1 “Søg efter tickets”
- should render search form
- should render text for loading
- should render one row in datatable

Følgende test er blevet opstillet:

- search view should see and validate search options
- search view should search for tickets assigned to `admin`

Sprint 6

Sprint planning

Denne sprint bliver planlagt anderledes end vi først havde forventet. Som tidligere beskrevet har vi indirekte fået udviklet den user story som havde planlagt release i sprint 6. Derfor går vi videre med user story 3. Dette er en beslutning som blev taget ved sidste sprint review og som nu er blevet bekræftet af product owner på trods af at han ikke selv kunne deltage ved sprint 5's review.

Han syntes det var en fornuftig beslutning på trods af at ikke alle acceptance kriterier var opfyldt. Hele funktionaliteten ved at kunne se en historik er implementeret og derfor har product owner allerede fået en hel del business value ud af denne. User story 5 vil stadig stå på product backloggen så der kan tages højde for acceptance kriterierne i en senere sprint.

Da user story 3 er estimeret til kun 2 story points har vi med i planlægningen prioriteret tid til flere tests. Dette er ikke noget vi normalt ville have gjort da det er noget som vi altid ville gøre løbende. Da dette er et skole projekt og dette er vores sidste planlagte sprint vil vi dog gerne afslutte på en god måde og være sikre på at vores kode er gennemtestet og gør som vi ønsker.

Nedenfor kan man se en tabel med planlægningen for sprint 6 med user story 3 i stedet for user story 5.

Ugeplan Sprint 6	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
Opaver:	Scrum møde Sprint planning Organisere sprint backlog. Tasks oversigt og tildeling	Påbegynde udvikling af user story 3's tasks.	Scrum møde Arbejde med Tasks	Arbejde med Tests hvis user story 3 er færdig	Scrum møde Arbejde med Tests hvis user story 3 er færdig Sprint Review

					Bruger test
					Retrospective

Sprint Backlog

Sprint backloggen blev udarbejdet på sædvanligt vis ved at dele den pågældende user story op i tasks.

User stories	Tasks	In progress	Done
US 3: Som supporter vil jeg kunne ændre status på tickets løbende så jeg kan se hvor mange der mangler handling.	implementer ændre status i angular frontend komponent implementer ændre prioritet i angular frontend komponent implementer status skift i backend implementer status prioritets skift i backend		

Opsummering af scrum møder

Mandag

Mandagens scrum møde blev relativt kort. Teamet var helt med på hvad dagen bød på af opgaver da vi har prøvet det nogle gange før.

Onsdag

Da teamet sammen havde udarbejdet user story 3 blev der holdt en hurtig opsummering af det færdige arbejde. Teamet regnede med at være klar til sprint review allerede nogle timer inde i onsdagens arbejde. Dette passer også fint overens med user storiens estimerede størrelse og teamets velocity. Der blev derfor aftalt at vi ville fortsætte arbejdet som planlagt med tests efter endt udvikling af user story 3.

Fredag

Fredagens scrum møde blev primært brugt til at kommunikere hvordan resten af dagen skulle gå. Desværre havde det endnu en gang vist sig at product owner ikke havde mulighed for at deltage ved det planlagte sprint review. For at kompensere for den manglende feedback, vil vi bruge 3 forskellige test personer i vores brugertest denne gang.

Sprint review

Som standin for product owner sad Andreas med ved dette sprint review og forholdte sig så objektivt til arbejdet som muligt. Da der i denne sprint er udviklet en meget lille user story er testen af produktet meget hurtigt overstået. Rent visuelt er user storien kun en drop down liste med forskellige statusser til en ticket. Efter at have ændret status på en ticket kan product owner se at den også bliver ændret i det view hvor man kan se alle tickets. Ud fra dette er acceptance kriteriet opfyldt og user storien kan markeres som done.

Herefter ville der normalt vis i et sprint review have været noget snak omkring product owners ønske for næste sprint. Da dette er et skole projekt og vi ikke udfører flere sprints var dette ikke et emne der blev berørt denne gang.

Som skrevet tidligere i rapporten er det svært at forholde sig helt objektivt til et stykke arbejde man selv har været med til at udvikle. Derfor er det også problematisk at teamet ikke får feedback fra den rigtige produkt owner.

Heldigvis har teamet som compensation for manglende product owner benyttet sig af brugertests som giver god feedback. Denne feedback er super brugbart da vi får meninger og synspunkter fra andre IT-folk som i sidste ende kunne være slutbrugere.

Brugertest

Til denne sidste brugertest har teamet valgt at lade tre forskellige testpersoner tage testen i stedet for en. Dette burde resultere i mere varieret feedback og måske i sidste ende, noget der kan gøre det til et bedre produkt.

Denne gang blev testpersonerne stillet en række nye opgaver som var opstillet ud fra de user stories som er blevet implementeret i programmet indtil nu. På den måde er vi sikre på at alt hvad vi har lavet bliver testet og at testpersonerne kommer rundt i hele programmet.

Som tidligere skrevet har programmet kun en hvis størrelse og det er med til at brugertesten er hurtigt overstået. Efter at have gennemført testen blev testpersonerne bedt om at komme med feedback.

Fælles for testpersonerne var at applikationen som helhed var utroligt fyldestgørende i forhold til det, som var formålet med produktet. Men det var heldigvis ikke kun ros teamet tog med sig fra mødet. Der kom også noget god konstruktiv feedback som absolut er værd at give videre til produkt owner så han kan overveje om det er noget han vil have med i programmet.

Det første vi fik som feedback var at søge funktionen skulle være mere synlig. To af de adspurgte testpersoner var enige i denne feedback. Det drejede sig om synliggørelsen af "Søg" som et punkt i navigations menuen. Fælles for disse to testpersoner var at de forsøgte at finde en bestemt ticket med et givent ID ved hjælp af sorterings funktionen inde i ticket viewet. Sorterings funktionen er ikke optimeret til dette og derfor brugte de meget tid på at løse opgaven før at de lagde mærke til at der var en "Søg" knap at navigere til.

Problemet kan løses på to måder. Man kan enten som testpersonerne ønsker gøre knappen mere synlig og navngive den anderledes, eller man kan ved udgivelse af produktet vedlægge en to sidet opstartsguide som kort præsenterer brugerne for de forskellige funktioner. Testpersonerne gav også udtryk for at når først de kendte til funktionen gav det iøvrigt fint mening at den lå sammen med alle andre navigationspunkter, men de var bare aldrig blevet præsenteret for den.

Det andet feedback vi fik, er noget som teamet også selv har været inde over. Det drejer sig om navngivningen af funktionerne: at skrive interne noter til en ticket og sende en mail som response til en kunde igennem en ticket. Som det så ud på test tidspunktet, markerede testpersonen i to "radio buttons" om det enten var "Intern" eller "Ekstern" besked.

Navngivningen af disse to er i sig selv ikke særlig beskrivende og derfor overvejer teamet nu at navngive dem til "Intern note" og "Mail svar" i stedet. Dette er dog noget der hurtigt kan rettes uden afsættelse af de store ressourcer.

Alt i alt en god brugertest med god feedback og dialog omkring produktet i dets nuværende stadie.

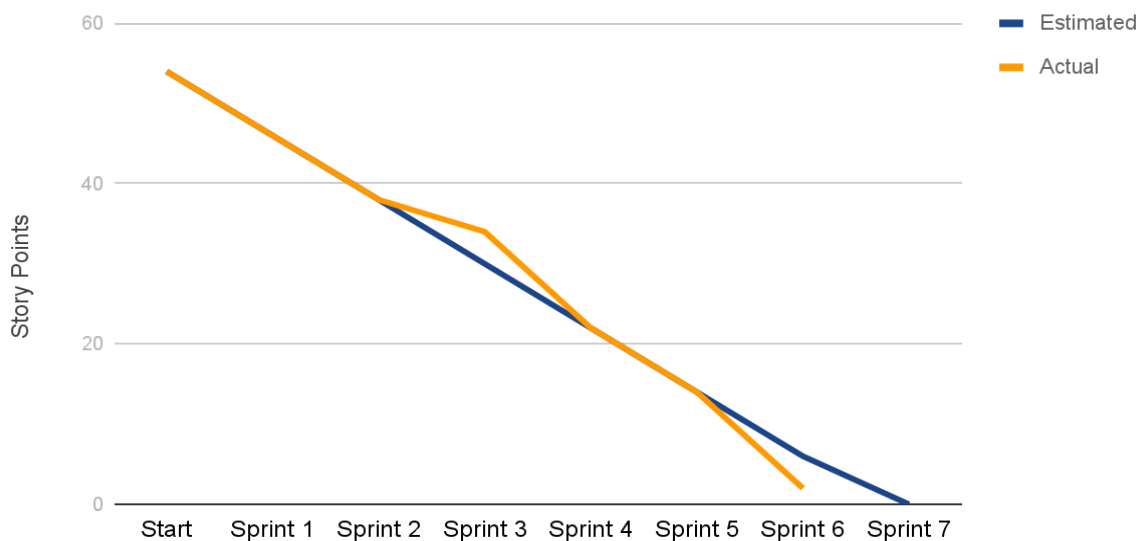
Burndown chart

Som noget af det sidste i sprint 6 gjorde vi igen status på vores burndown chart. På denne sprints burndown chart kan man se at teamet er lidt foran. Dette er et resultat af at en user story som er estimeret til 8 story points indirekte næsten er blevet implementeret i

forbindelse med udviklingen af andre user stories. Vi ender derfor med 2 story points tilbage på projektet i sprint 6, da acceptance kriterierne ikke helt er opfyldt for user story 5.

Burndown Chart

End of Sprint 2



Sprint retrospective

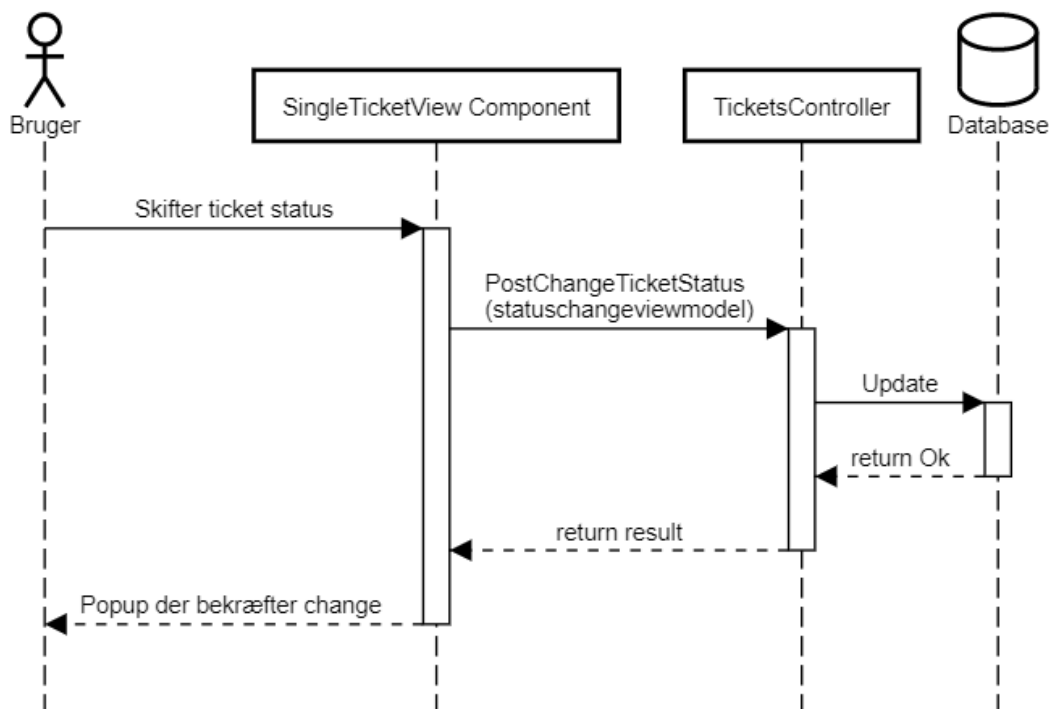
Til denne sidste sprints retrospective er det generel enighed om, at i sprint 6 har arbejdsmoralen været høj. Teamet har været motiveret for at komme i mål på en god måde og det smitter af på arbejdet.

Derudover har kommunikationen været rigtig god i forbindelse med de tests som vi ville have afviklet i denne sprint. Når der var researched nogle ting som vi ikke var så gode til var researcheren god til at videregive information og "undervise" teamet så vi var så effektive som muligt. Man kan mærke at teamet har været igennem nogle sprints sammen nu og det gør at arbejdet bliver mere flydende uden så mange spørgsmål til hvordan vi skal gøre tingene.

Tekniske detaljer

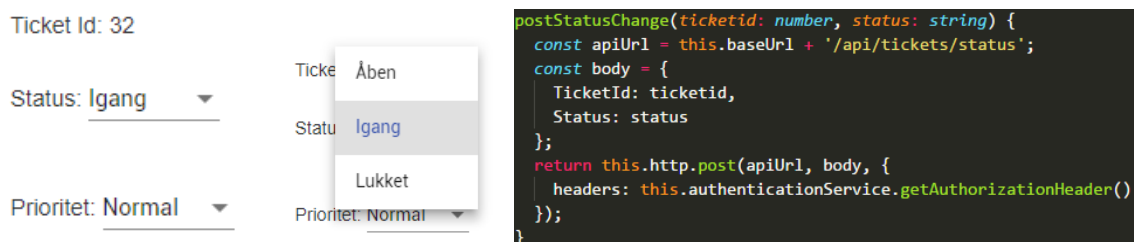
User story 3 består af fire små tasks, som vil beskrives forholdsvis som frontend og backend. User historien går ud på at, man skal kunne ændre status på en ticket, det var dog også tænkt på at man skulle kunne ændrer på prioriteten, da ordet status også beskrev prioritet som en status, så indholdet af user story 3's tasks blev udvidet til de fire tasks den består af. Herunder kan der visuelt ses, hvad user story 3 skal formå at gøre:

User story 3



Status skift i view componentet

Da singleticket view componentet først blev implementeret, var det uden mulighed for at skifte status, dette har vi dog løst ved at bruge “mat-select”³⁰ fra material angular. Her har vi valgt at kalde en funktion som modtager en “MatSelectChange” event som argument, hvilket gør arbejdet for at finde den valgte værdi nemmere.



Når systemet fanger en af disse “mat-select”s skifter værdi, kalder den en tilsvarende funktion på componentet. Denne funktion får en service til at udføre et http kald, mod et endpoint i backenden, hvorefter angular giver en tilbagemelding ved success i form af en snackbar popup besked. Både det at skifte status og skifte prioritet har en meget ens implementation, da det er meget ens, dog kunne der godt have været udtænkt en mere generisk implementation, som tiltænkt på diagrammet i starten af denne tekniske detaljer.

³⁰ <https://material.angular.io/components/select/overview>

Status skift endpoints

Ideen var at der skulle være en log, over hvilke handlinger brugerne af systemet foretager sig, så derfor blev http endpointet designet som en POST. Da vi er i den sidste sprint, og vi ikke havde planlagt at gøre det mere avanceret end højst nødvendigt, valgte vi at holde det simpelt med hensyn til vores REST design. implementeringen er meget simpel, som bare går ud på at finde den ticket som er refereret i view modellen, hvor man efterfølgende skifter status og tilbakemelder at dette blev gjort.

```
/// <summary>
/// POST: api/Tickets/status
/// </summary>
/// <returns>Returns result if status changed</returns>
[HttpPost("status")]
0 references | Bang0123, 3 hours ago | 1 author, 2 changes | 0 requests | 0 exceptions
public async Task<IActionResult> PostChangeTicketStatus(
    [FromBody] TicketStatusChangeViewModel change
) {...}

/// <summary>
/// POST: api/Tickets/priority
/// </summary>
/// <returns>Returns result if priority changed</returns>
[HttpPost("priority")]
0 references | Bang0123, 3 hours ago | 1 author, 2 changes | 0 requests | 0 exceptions
public async Task<IActionResult> PostChangeTicketPriority(
    [FromBody] TicketPriorityChangeViewModel change
) {...}
```

Testing

Med denne user story har det ikke givet meget mening at prøve at unit teste, da vi interagerer med vores backend, vi kunne til fordel have benyttet integrations tests, men det har igen vist sig ikke at være nemt at implementere.

I dette afsnit med tests, implementeres der alle de automatiske acceptance tests, vi har opstillet i forrige sprints. Derudover skal der også opstilles og implementere acceptance tests for dette afsnit.

Acceptance tests

De automatiske tests, som lige nævnt i sprint 1 afsnit om testing, benytter protractor³¹ og en instans af selenium webdriver. Vi har implementeret alle acceptance tests fra forrige sprints, og herunder ses et billede af nogle e2e tests fra tickets viewet, hvilket kan give et indtryk af den generelle implementering:

³¹ <http://www.protractortest.org/#/>

```
it('should see open tickets', () => {
  expect(ticketspage.getOpenTicketsTextPresent()).toBeTruthy();
  expect(ticketspage.getOpenTicketsNumberPresent()).toBeTruthy();
});

it('should see critical tickets', () => {
  expect(ticketspage.getCriticalTicketsTextPresent()).toBeTruthy();
  expect(ticketspage.getCriticalTicketsNumberPresent()).toBeTruthy();
});

it('should see tickets in All tickets datatable', () => {
  expect(apppage.getTicketDataTableRowsPresent()).toBeTruthy();
});

it('should change filter of tickets and back', () => {
  expect(ticketspage.testFiltering()).toBeTruthy();
});

import { browser, by, element } from 'protractor';
export class TicketsPage {
  getOpenTicketsTextPresent() {
    return element(by.css('div[name="openticks"] h3'))
      .isDisplayed();
  }

  getOpenTicketsNumberPresent() {
    return element(by.css('div[name="openticks"] p'))
      .isDisplayed();
  }

  getCriticalTicketsTextPresent() {
    return element(by.css('div[name="criticalticks"] h3'))
      .isDisplayed();
  }

  getCriticalTicketsNumberPresent() {
    return element(by.css('div[name="criticalticks"] p'))
      .isDisplayed();
  }

  testFiltering() {
    const headerBtn = element(by.css('mat-table mat-header-row mat-header-cell div button'));
    const statustxt = element(by.css('mat-table mat-row mat-cell')).getText();
    headerBtn.click();
    const statustxt2 = element(by.css('mat-table mat-row mat-cell')).getText();
    if (statustxt !== statustxt2) {
      headerBtn.click();
      return true;
    } else {
      headerBtn.click();
      return false;
    }
  }
}
```

Resultat fra e2e tests:

```
Executed 46 of 46 specs SUCCESS in 33 secs.
[20:34:10] I/launcher - 0 instance(s) of WebDriver still running
[20:34:10] I/launcher - chrome #01 passed
```

Det løb op i alt 46 automatiske acceptance tests, på nuværende tidspunkt, da vi endnu ikke har opstillet testene for user story 3. Vi har valgt at opstille to tests:

- ticket view should render snackbar popup on status change
- ticket view should render snackbar popup on priority change

```
it('should render snackbar popup on status change', () => {  
  singlepage.testChangeStatus();  
  expect(apppage.getSnackBarText()).toContain('Status Changed');  
});  
  
it('should render snackbar popup on priority change', () => {  
  singlepage.testChangePriority();  
  expect(apppage.getSnackBarText()).toContain('Priority Changed');  
});  
Executed 48 of 48 specs SUCCESS in 43 secs.  
[13:27:50] I/launcher - 0 instance(s) of WebDriver still running  
[13:27:50] I/launcher - chrome #01 passed
```

Herover ses resultatet af de sidste to opstillede automatiske acceptance tests, tiden de 2 har tilføjet ekstra skyldes de snackbars, som skal popup, ikke er fantastiske til automatiserede tests, derfor steg test tiden fra 33 sekunder til 43 sekunder.

Konklusion

Ved starten af projektet opstillede vi en problemformulering som vi ønskede at være i stand til at besvare når vi var færdige med projektet. Det har været en lærerig oplevelse og de erfaringer vi har gjort os løbende, har hjulpet os med at vokse som softwareudviklere. Derfor føler vi nu at vi kan besvare dem på en fyldestgørende måde.

For at besvare det første spørgsmål har vi måttet bruge flere teknologier. Men hovedparten af arbejdet for at få et overblik kan deles op i tre primære funktioner for løsningen.

Første funktion er at web applikationen skal kunne hente data omkring hvilke supportsager den pågældende virksomhed ligger inde med.

Anden funktion omhandler at arrangere den indhentede data på en måde så det er let at læse og nemt at overskue.

Den tredje og sidste primære funktion er at virksomheden skal kunne arbejde med de supportsager som nu er arrangeret på en brugervenlig og effektiv måde.

I vores løsning valgte vi at løse den første primære funktion ved at have et baggrundsjob kørende i programmet som indhentede mails da det var i en mail indbakke som vores product owner havde alle hans sager liggende i. Den anden primære funktion løste vi ved at benytte material angular og bootstrap til at lave en praktisk brugerflade som var brugervenlig og minimalistisk. Den tredje primære funktion har vi løst ved at give brugeren en masse funktionalitet hvor han blandt andet kan svare på support sager, skriver noter til dem og ændre deres status. Det er med disse tre primære funktionaliteter i tankerne, at det har lykkedes os at lave en web applikation som kan hjælpe en virksomhed med at få overblik over sine supportsager.

I løbet af opgaven har vi forsøgt at benytte os af teori fra undervisningen til at hjælpe os med at styre projektet. De gange hvor vi har stået over for nogle problemstillinger, har vi indset at vi havde nogle gode redskaber fra undervisningen, som kunne hjælpe os med at komme videre. Disse værktøjer er ikke nødvendigvis noget som kommer fra scrum eller XP. I virkeligheden handler det om at forstå hvornår de forskellige teorier og modeller kan hjælpe dig med at løse den problemstilling du står overfor. Foruden at benytte sig af modeller og methodologies kan vi også konkludere at kommunikation med din product owner er en vigtig del af processen. Hvis man kan opretholde en god to-vejs kommunikation med product owner og formår at mestre brugen af softwareudviklings teori i hverdagen er vi sikre på at man nok skal komme frem til et resultat som begge parter er glade for.

Refleksion

Foruden scrum møderne, som vi kommer tilbage til senere, var det første i vores sprints selve planlægningen. Planlægningen gjorde at vi fik diskuteret hvordan vi skulle opnå målet for hvert enkelt sprint. I vores projekt har vi nået målet for vores sprints næsten hver gang og derfor har det ikke været alt for udfordrende for os at følge den grove planlægning der blev lavet på den første version af product backloggen. Vi kunne mere eller mindre følge de planlagte releases der var blevet estimeret med. Derudover brugte vi vores sprint planlægninger til at opdele den pågældende sprints user story op i tasks. Størrelsen på de tasks som vi fik delt op i passede rigtig godt for os i størrelse og gjorde også at det var lettere at få overblik over arbejdet der lå foran os.

Vores formål med sprint planlægningen var i bund og grund at teamet forstod hvad ugens arbejde bød på og hvordan vi skulle komme i mål. Dette følte vi at vi lykkedes med og betegner derfor også vores sprint planlægninger som en success.

I løbet af vores sprints og som arbejdet skred frem afholdte teamet også scrum møder. Dette gjorde vi tre gange i hver sprint. Vi tror at mængden af scrum møderne var passende. At holde scrum møde hver dag havde ikke bidraget med noget til projektet og omvendt så havde vi risikeret at der ville være for lang tid imellem møderne hvis vi havde afholdt færre. Længden på vores scrum møder varierede meget. Nogle af vores scrum møder var korte og effektive i den forstand at de blev brugt som en hurtig "status opdatering" på det udførte arbejde. Andre trak ud og blev nærmest til en mere teknisk gennemgang af kode for at alle parter af teamet var med på hvordan koden virkede. Dette fungerede godt for os da det er et skole projekt men vi tror ikke at det vil være en effektiv måde at afholde møderne på ude på arbejdsmarkedet.

Efterhånden som vi kom til slutningen af hver sprint afholdte vi også et sprint review. Sprint reviewet var i de tilfælde hvor den rigtige product owner deltog en meget lærerig aktivitet. Det var en god oplevelse, at føle man udviklede produktet for nogen og motiverede os til at gøre os mere umage, end i eksempelvis tidligere projekter på uddannelsen. Når det så er sagt følte vi ikke helt at vi fik det maksimale ud af vores sprint reviews i de tilfælde hvor det var en fra gruppen der sad stand in som product owner. Vi fik det til at fungere men som vi har givet udtryk for i rapporten var det svært at holde sig objektivt til produktet og se fejl/mangler som en product owner måske havde set.

For at gøre op for dette besluttede vi os for i hver anden sprint at afvikle en brugertest. Brugertesten var nok en af de bedste beslutninger vi tog i forbindelse med feedback til det produkt som vi var i gang med at udvikle. Det var fedt at se hvordan testpersonerne gik til opgaven og se dem navigere i den applikation som vi havde lavet. Det gav os også et rigtig godt indblik i hvor den kunne forbedres. Den eneste negative ting ved brugertesten var at forberedelsen af opgaverne måske kunne have været bedre. Nogle gange gjorde formuleringen af vores opgaver til de adspurgte testpersoner at de ikke var helt sikre på hvordan de skulle komme til løsningen. De kom dog alle igennem og vi havde efterfølgende en god dialog omkring hvad de lige havde oplevet. Dialogen og feedback omkring applikationen var formålet med brugertesten og det er bestemt noget vi kunne se os tage med videre ud på arbejdsmarkedet.

Efterhånden som vi blev færdige med de forskellige sprints havde vi afsat tid til to sidste ting i dem. Den ene var at udarbejde et burndown chart. Burndown chartet var en del af rapporten som vi begge fra start af gerne ville have med. Desværre så føler vi ikke at burndown chartet bidragede med specielt meget til vores arbejdsgang og kvaliteten af produktet. Vi er meget enige om at det er vigtigt at holde status med om man overholder den velocity man har planlagt med, men på en skala som dette projekt befinder sig på er det ikke svært at holde trit med dette. Der har trods alt kun være to udviklere og vores planlagte velocity har været 8 story points for hver sprint.

Når det så er sagt tror vi at burndown chartet har sin plads i større projekter med flere teams og mangle flere user stories. Vi har dog valgt at beholde det i vores rapport da der har været brugt tid på det og det er en fin visuel illustration af hvordan vi har arbejdet.

Den anden ting som vi afsluttede hver sprint med var vores retrospective. Vores retrospective har været en positiv oplevelse for os begge. Vi brugte det som et frirum hvor man kunne tale frit omkring hvordan man selv har bidraget til opgaven men også hvilke observationer man har gjort sig ved teamet som en helhed. Der var plads til at komme med forslag til hvad vi kunne gøre anderledes uden at der blev rynket på næsen. Nu har teamet heldigvis været enige omkring det meste i løbet af hele projektet og der har været høj moral hele vejen. Vi føler at der er mange der undervurderer et sådan møde og ikke ser vigtigheden i at åbne dialog omkring hvordan man arbejder sammen som et team. Dette indebærer bestemt ikke kun de dårlige ting men også hvilke ting der har fungeret godt som teamet gerne skulle fortsætte med.

For rapportens skyld har vi som det sidste inkluderet nogle tekniske detaljer for at vise hvordan vi rent teknisk er kommet frem til det produkt som vi har i dag. Det endte med at

blive en god ting for os selv også da det var en god mulighed for at have en mere teknisk gennemgang af koden.

Alt i alt et projekt som vi er glade for og stolte af.

***Tak** til Bo fra synergj for hans tid og lyst til at deltage i dette projekt.*

*Og ikke mindst **Tak** til Jamshid Eftekhari for gode råd og vejledning.*

Litteraturliste

Websites

<http://www.scrumguides.org/scrum-guide.html#purpose> - scrum guide

<http://www.scaledagileframework.com/spikes/> - tekniske spikes

<http://www.mimekit.net/> - fully featured Mail framework for .NET og .NET Core

<https://www.hangfire.io/> - Hangfire for .NET og .NET Core

<https://blogs.msdn.microsoft.com/dotnet/2017/08/14/announcing-net-core-2-0/> - .NET core

2.0 announcement page

<https://angular.io/> - Angular 4/5 website

<https://docs.microsoft.com/en-us/aspnet/core/data/ef-mvc/migrations> - intro to migrations on msdn

<https://material.angular.io/> - google's material theme and cool components

<https://karma-runner.github.io/2.0/index.html> - Karma test runner

<https://jasmine.github.io/> - Jasmine test suites

<http://www.protractortest.org/#/> - Protractor for automatic e2e tests

<http://docs.identityserver.io/en/release/> - Identityserver4 docs website

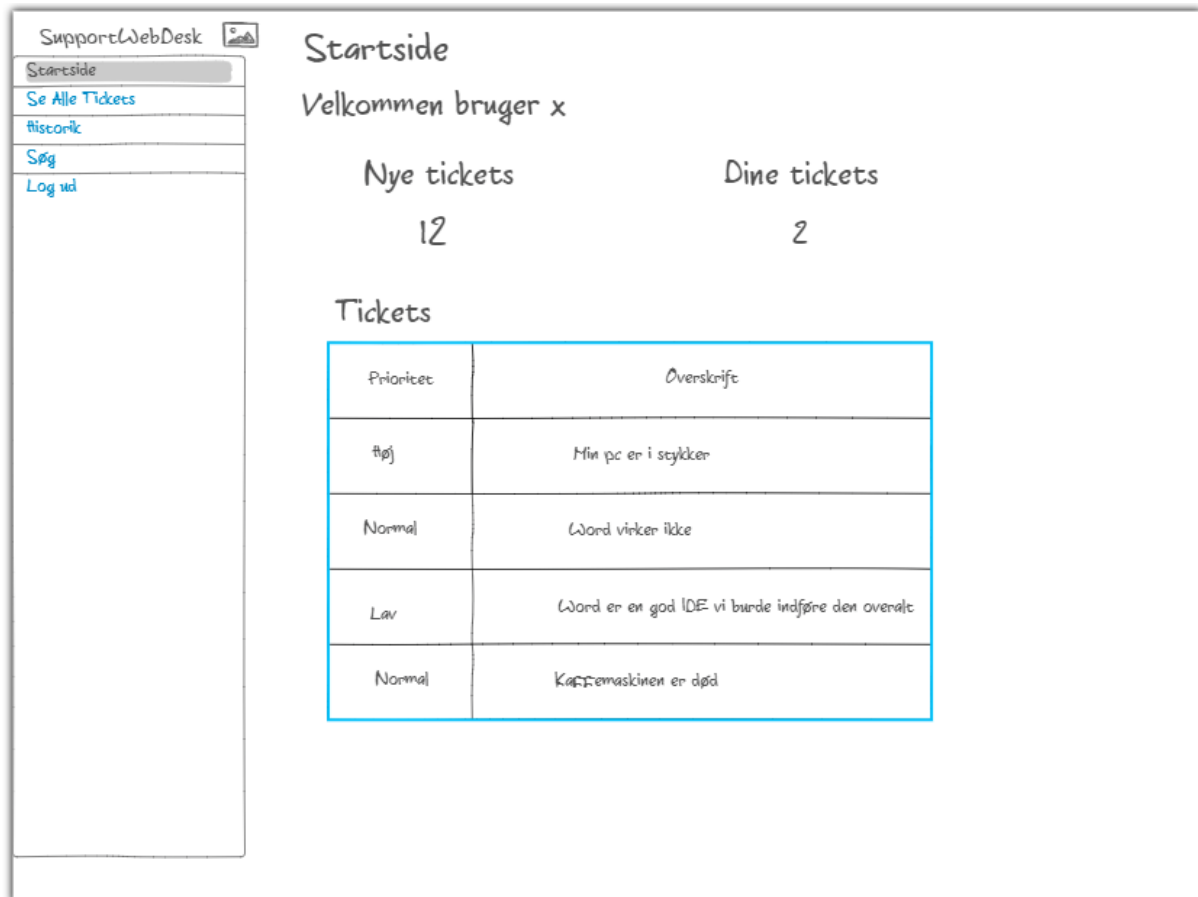
<https://github.com/manfredsteyer/angular-oauth2-oidc> - Angular Oauth2 oidc libs

<http://www.albahari.com/nutshell/linqkit.aspx> - Linqkit, predicatebuilder website and libs


Bilag

Prototypes

Bilag 1.1



Bilag 1.2

SupportWebDesk 

[Startside](#)
[Se Alle Tickets](#)
[Historik](#)
[Søg](#)
[Log ud](#)

Løste tickets

Løste tickets
1337

Status	Prioritet	Overskrift	Anmoder	Ansvarshavende	Sidst opdateret
Lukket	Høj	Hjælp mig	Jørgen Spønsen lvs	Agent 1	02-11-2017 13:37
Lukket	Normal	Hjælp Jørgen	Hans søn	Agent 1	02-11-2017 13:37
Lukket	Kritisk	Word virker ikke	Person	Agent 2	02-11-2017 13:37
Lukket	Normal	serveren er crashed	person	Agent 3	02-11-2017 13:37
Lukket	Normal	Min browser lukker	person	Agent 2	02-11-2017 13:37
Lukket	Normal	Lorte service	person	Agent 1	02-11-2017 13:37
Lukket	Normal	Jeg har ikke signal	person	Agent 5	08-10-2017 13:37
Lukket	Normal	Intet problem	person	Agent 1	02-10-2017 13:37
Lukket	Normal	min ven jeg er sulten	person	Agent 4	01-11-2017 13:37
Lukket	Normal	Durum?	person	Agent 1	01-11-2017 13:37

Bilag 1.3

SupportWebDesk

[Startside](#)
[Se Alle Tickets](#)
[Historik](#)
[Søg](#)
[Log ud](#)

Ticket

Min pc er i stykker

Gå til seneste svar i tråden

Anmoder

Ansvarshavende

Anmoder

hjælp mig min pc er i stykker

mvh Jørgen

Høj Jørgen

Har du prøvet at genstarte?

mvh support

Høj support

jeg har stadig problemet

mvh Jørgen

Svar felt

☒ Intern
☐ Ekstern

Send

Dato

ticket id

prioritet

Ansvarshavende

Anmoder

Mine Jørgen Sørensen

Relaterede tickets

1 stykker

hehe

noobs

interne noter

Han er en rar person

Han er en ikke så god

82

Bilag 2

Shared Vision

Support Web Desk

1. Introduction

Synergi har en mailbox der agere support lige pt, dette giver dårligt overblik over hvornår en kunde skal have hjælp og hvornår de har efterspurgt hjælp. Derfor ønsker Synergi et support system som kan give overblik.

2. Positioning

2.1 Problem Statement

Problemet:	Uoverskuelig kundesupport.
Påvirker:	Synergi og deres kunder.
Problemet gør at:	Mere unødvendigt søge arbejde og tabt tid.
En god løsning ville være:	Mere overskuelig løsning, der giver overblik over support sager.

2.2 Product Position Statement

Til:	Synergi.
Hvem:	Kunde support.
Support Web Desk	Et redskab til organisering af kundehenvendelser.
Som:	Hjælper Synergi med at holde overblik over deres support sager.
Unlike	Store kommercielle support systemer som eks. Zendesk.
Vores produkt:	Simpelt system der giver et hurtigt overblik over support sager.

3. Stakeholder Descriptions

3.1 Stakeholder Summary

Name	Description	Responsibilities
Synergi	Dem der hjælper kunder med support.	At få afviklet efterspørgslen på support hurtigst muligt.

3.2 User Environment

En web applikation der virker på flere platforme, skal kunne give overblik over flere henvendelser på en gang.

4. Product Overview

4.1 Needs and Features

Need	Priority	Features
Brugersystem	2	Login ved brug af E-mail og password.
Support sags noter.	5	Skrive noter til alle supportsager løbende.
Support sager view.	1	Se alle support sager.
Historik.	7	Se historik over kunde sag.
Søge en support sag.	6	Søg efter specifik support sag.
Autosvar.	4	Efter support henvendelse sendes svar retur.
Svar på support sager.	3	Mulighed for at svare kunden på en support sag.

5. Other Product Requirements

Requirement
<ul style="list-style-type: none">- Usability <p>Brugen af systemet skal være nemt og let tilgængelig.</p> <p>Bruger interface skal være let at benytte.</p>
<ul style="list-style-type: none">• Portability <p>Web applikationen skal nemt kunne tilgås på flere platforme.</p> <p>Den skal være Restful, så der nemt kunne laves en native App til IOS, Android, Windows eller Mac.</p>
<ul style="list-style-type: none">• Fault tolerance- Systemet skal helst kunne køre videre med dens hjertesag, selvom der opstår fejl i andre mindre dele af systemet.

Bilag 3

Product Backlog

Nr.	Prioritet	Beskrivelse/business value	Acceptance kriterier	Release
1	2	Som supporter vil jeg kunne logge ind og se mine tickets så jeg har overblik over hvem der mangler min hjælp. Business value: 85	<ul style="list-style-type: none">- Antal af tickets som afventer handling fra aktuelle bruger vises på forside.	Begynder udvikling: sprint 1 Forventet release: Sprint 3
2	4	Som supporter vil jeg kunne skrive noter til tickets løbende så jeg har overblik over alle sagens tiltag. Business value: 65	<ul style="list-style-type: none">- Noter tilføjes til den pågældende tickets historik.- Noter kan læses af alle supportere.- Noter gemmes i database.	Begynder udvikling: sprint 4 Forventet release: Sprint 5
3	7	Som supporter vil jeg kunne ændre status på tickets løbende så jeg kan se hvor mange der mangler handling. Business value: 35	<ul style="list-style-type: none">- Status skal ændres i hele systemet umiddelbart efter ændring er udført.	Begynder udvikling: sprint 7 Forventet release: Sprint 7
4	1	Som supporter vil jeg kunne se alle support sager så jeg har overblik over hvor mange tickets vi har åbne. Business value: 100	<ul style="list-style-type: none">- Alle tickets skal vises på en liste.- Det skal være muligt at ændre på sorteringen af tickets	Begynder udvikling: sprint 1 Forventet release: Sprint 1

			(eks. status).	
			<ul style="list-style-type: none"> - Det skal være muligt at klikke på en ticket for at åbne den. 	
5	6	<p>Som supporter vil jeg kunne se en tickets historik så jeg har overblik over hvilke tiltag der tidligere er blevet gjort.</p> <p>Business value: 45</p>	<ul style="list-style-type: none"> - På en tickets historik skal supporter kunne se andre supporters tiltag og noter. - Der skal være datostempel på alle handlinger i historikken. - Der skal stå hvilken support bruger der har udført handlingen eller skrevet noten. 	<p>Begynder udvikling: sprint 6</p> <p>Forventet release: Sprint 7</p>
6	5	<p>Som supporter vil jeg kunne søge efter tickets så jeg hurtigt kan finde en support sag.</p> <p>Business value: 60</p>	<ul style="list-style-type: none"> - Der skal kunne søges på en ticket ud fra navn på kunde. - Der skal kunne søges på en ticket ud fra kundens firmanavn. - Der skal kunne søges på en ticket ud fra ticket Id. 	<p>Begynder udvikling: sprint 5</p> <p>Forventet release: Sprint 6</p>

			<ul style="list-style-type: none"> - Der skal kunne søges på en ticket ud fra overskrift. 	
7	3	<p>Som supporter vil jeg kunne svare tilbage på supportsager inde i den pågældende ticket så jeg ikke lader kunder vente i uvished.</p> <p>Business value: 80</p>	<ul style="list-style-type: none"> - Der skal kunne skrives en tekst i et felt i den pågældende ticket som sendes via mail til kunden. - Systemet skal automatisk indsætte en mail signatur som er gemt til bruger. - I mailen der sendes til kunde skal ticket id være inkluderet i mailens emnefelt. 	<p>Begynder udvikling: sprint 3</p> <p>Forventet release: Sprint 5</p>

Bilag 4

Trello som backlog



Bilag 5

Support Web Desk

[Startside](#)

[Tickets](#)

[Ticket](#)

[Søg efter tickets](#)

[Logout](#)

Søg efter tickets

Ticket id

32

Overskrift

Prioritet

Dato fra

Søg

Anmoder

Fri text

Ansvarshave...

Dato til

Filter tickets

Status	Prioritet	Overskrift	Anmoder	Agent	Sidst opdateret	Se Ticket
Igang	Normal	Hej	Nikolaj Bang	admin@gmail.com	15:20 07-12-2017	Se Ticket

Items per page: 5 1 - 1 of 1 < >