EA Dania Skive

3. Semester – SK16DASX116

Systemudviklings Eksamen

Vejledere:

Klaus Nørregaard
Christian Clausen

Link til BudgetManager:
budgetmanagerxenaeksamen.azurewebsites.net

Login til Xena:
Skovmissen@gmail.com
Dinmor123

BUDGETMANAGER

Nikolaj Skovmose – Lasse Meldgaard – Patrick Rechnitzer – Anders Oxenvad

1 Indledning

I denne rapport vil vi komme ind på processen, for hvordan vi har udviklet vores budgetmanager.

Denne budgetmanager vil vi lave sådan at kunder hos Xena vil kunne sammenligne deres inddaterede budget sammen med det regnskab som virksomheden har inde på Xena.

Vi har gennem projektet anvendt metoder og modeller som vi er blevet undervist i på i studiets 1, 2 og 3 semester.

Det afleverede projekt give et godt resultat hvad vi som gruppe kan levere på 3 uger. En løsning som vi vil vurdere, til at opfylde de formelle krav til funktionaliteten som er blevet stillet. Dertil har vi gennem hele projektet udførligt dokumenteret, hvordan vi har brugt relevante analyser og modeller.

Denne rapport er udformet så den følger den udviklingsproces vi er gået igennem, og som vores produkt er blevet udformet. Dertil kan man ved at gennemse indholdsfortegnelsen, hvordan hele projektet er tænkt og struktureret.

God læselyst.

2 Projekt opstart

Af: Anders

Vi startede vores projektperiode mandag den 27-11 hvor vores 'Product owner' Klaus Nørregaard holdte oplæg om hvad vores produkt skulle indeholde samt hvilke retningslinjer vi skulle holde os indenfor. Der var mulighed for at stille spørgsmål til både Klaus Nørregaard og Christian Clausen.

Efter oplægget satte vores gruppe sig ned sammen og læste opgaveformuleringen igennem en ekstra gang. Vi satte os hver især ned og skrev hvad vi mente der var vigtigt at fokusere på, derefter diskuterede vi hvad alle i gruppen havde noteret.

Vi prøvede at finde et lokale, som vi kunne have for os selv med et stille miljø, dette var desværre ikke muligt på første dagen.
Vi var alle enige om at vi ville møde op på studiet hver dag, da vi mente vi ville arbejde bedst der og have muligheden for diskutere indbyrdes mellem hinanden, dette ville også sikre sig at man som individuel gruppe medlem, ville arbejde så længe man var på studiet. Vi ville også have muligheden for at kunne kontakte vores 'product owner' når vi har noget vi vil præsentere for ham.
Vi lavede derefter, samlet hele gruppen, vores domæne model for at sikre at alle var enige omkring udformningen af denne.
Efterfølgende begyndte vi at skrive user stories. Vi valgte at bruge 'planning poker' til at estimere hvor lang tid der skulle bruges til hver user story. Efter det fik vi lavet task til vores user stories, hvor vi igen brugte 'planning poker' til at estimere tiden.

Vi sluttede første dag af med at finde nogle tasks vi var sikker på vi kunne blive færdig med og så sørgede vi for at få dem sat over i 'done' på vores scrum board før vi måtte tage hjem.

Vi har fra start af forsøgt, at inddrage vores PO, når vi har haft spørgsmål til resultatet af produktet, så vi ikke blev nødsaget til at stoppe processen, grundet misforståelser.

2.1 Risikoplan

Af: Lasse

-Risikoanalyse tabel- Sandsynlighed 1-5 Konsekvens 1-3-7-10 Produkt = sandsynlighed * konsekvens							
Risiko Moment	Sand-synlig-hed	Kon-sek-vens	Pro-dukt	Præventive tiltag	Ansvarlig	Løsnings forslag	Ansvarlig
Tidspres	3	3	9	Lave en tidsplan i starten af projektet.	Gruppen	Overholde tidsplan, ellers er overarbejde påkrævet.	Gruppen
Manglende dokumentation på API fra Xena	3	3	9	Dokumentation er ikke noget vi kan påvirke.	Gruppen	Vælge en anden API, hvis muligt, eller kontakte Xena.	Gruppen
Sygdom	2	3	6	Sørge for en god gruppe dynamik, og undgå stressede situationer	Gruppen	Sørge for at få lavet noget hjemme, ved sygdom.	Gruppen
Frafald	1	7	7	Skabe god team dynamik Pas på hinanden	Gruppen	Sørg for at alle er sat ind i alt	Gruppen
Misforståelser/ Manglende forståelse af opgaven	3	3	9	Læse opgaven igennem mere end en gang. Få svar på alle tvivlsspørgsmål	Gruppen	Inddrage PO'en i de større beslutninger	Gruppen
Fejl Merge i versionshåndtering	3	3	9	Få sat projektet op i git	Gruppen	Ved eventuelle merge problemer, skal 2 i gruppen godkende mergen.	Gruppen
Manglende forventnings afstemning	2	3	6	Lave en forventningsaftale inden projektopstart	Gruppen	Ved uoverensstemmelser kan der kigges i kontrakten	Gruppen

Vi har lavet en risikoanalyse, for at være på forkant med eventuelle problemer der kan opstå i projektforløbet. Dette kan spare os tid på længere sigt, da vi både har lavet præventive tiltag men også løsningsforslag, hvis et risikomoment skulle opstå.

En af de risikomomenter der har det største produkt er *Misforståelser/Manglende forståelse af opgaven*, hvis vi arbejder hen imod en løsning, som vi i gruppen tror er rigtig, men det ikke er hvad kunden ønsker, kan der være mange timer tabt. Det er derfor vigtigt vi er på forkant med dette, og får taget kontakt ofte til PO'en for at sikre vi arbejder mod det rigtige resultat. Vi har ikke valgt nogen, der decideret har ansvaret for de forskellige risikomomenter, men derimod at vi samlet på gruppen er ansvarlig for at holde fokus på disse, da vi som gruppe har et samlet ansvar for hele projektet.

3 Mål

Af: Anders

Vores mål er at skabe en Budgetmanager, som skal være i stand til at synkronisere med Xena's budget og på den måde hente de eksisterende finansgrupper og finanskontoer ud fra Xena budgettet.

Det skal derefter være muligt ud fra de synkronisere grupper/kontoer at inddatere sit budget som ønsket og derefter kunne sammenligne sit budgetmanager budget med det regnskab som kunden har liggende på Xena.

Vi vil ved hjælp af "SCRUM" administrere vores opgaver/task ud fra de sprints vi har defineret på gruppen.

Vores mål for autentifikation er at bruge Oauth via Xena for at man skal kunne logge på budgetmanager, det har vi valgt for at sikre sikkerhed i vores budgetmanager.

Der skal udelukkende bruges Github som versionskontrol for at sikre alt er opdateret, og al data altid vil være tilgængelig for alle gruppens medlemmer.

4 Problemstilling

Xena.biz har ikke et dedikeret budgetværktøj til sine kunder. Kunder der ønsker at arbejde med budgetprocessen og budgetplanlægning, er nødsaget til at finde alternative værktøjer. Kontoplanen i Xena viser kun de tal, som man har indrapporteret i Xena i forbindelse med varekøb, forbrug, salg m.v. Kontoplanen er således et billede af historiske aktiviteter som virksomheden har udført. At Xena ikke har et budgetværktøj er utilstrækkeligt for de kunder der ønsker at handle på baggrund af en sammenstilling af budgetter og regnskab. En sammenstilling af budgetter og regnskaber vil kunne give brugerne af regnskabet et overblik over realiseret aktiviteter og de budgetterede – løbende og med en minimal forsinkelse.

4.1 Perspektivering af problemstilling

Af: Nikolaj

Der uddrages fra oplægget at Xenas brugere ikke har et budgetværktøj, og at vi som udviklere skal udarbejde en tredjeparts app (Budgetmanager) som kan implementeres på Xena.

Det er blevet beskrevet at brugerne skal kunne få et realistisk overblik over deres budget, ved hjælp af en sammenligning af de reelle tal fra deres regnskaber på Xena.

5 Problemformulering

Af: Alle

- Hvordan sikre vi at vores brugers data er beskyttet?
- Hvilke udfordringer opstår der ved at hente data ud fra Xenas API?
- På hvilken måde vil vi sammenligne data fra Xenas budget og de oprettede budgetter i vores Budgetmanager?
- Der kan opstå nogle udfordringer hvis dataene i de oprettede budgetters finanskonti ikke stemmer overens med de data der er i Xenas, på hvilken måde vil man løse dette?
- Hvordan kan vi give mulighed for at oprette et budget på en intuitiv måde for brugerne?

6 Krav til løsning

Af: Patrick

Hvilke krav stilles der til vores løsningsforslag? Hvad skal vores løsningsforslag kunne?

Kunden har opstillet nogle krav til hvad vores løsningsforslag skal indeholde. Kunden har nogle specifikke funktioner, som vores budgetmanager skal kunne for at opfylde kundens ønske til produktet. Kundens krav er:

- Oprette, redigere og slette budgetter
- Koble budgetter til regnskaber i Xena
- Se hvordan aktuelle konti forholder sig til de budgetterede konti

Ud fra kundens krav til budgetmanageren, skal det være muligt at koble budgetter sammen med regnskaber på Xena. Dette vil give kunden mulighed for at få et overblik over de kommende udgifter og indtægter i regnskabet.

7 SWOT

Af: Anders

7.1 Hvorfor bruger vi SWOT?

Vi har valgt at lave en SWOT analyse da den er med til at give et rigtig godt indblik i projektets, interne forhold, de svage og stærke sider. På den måde kan den hjælpe med at finde ud af hvad der måske trænger til forbedringer og hvor man måske ikke behøver ligge mange resurser.

Når vi kigger på projektets eksterne forhold, muligheder og trusler, så bruger vi disse for at finde de trusler som kommer udefra. Altså dem som vi ikke kan påvirke som udviklere, dette kunne f.eks. være en ny finanskrise der gjorde at virksomhederne ville være mere varsomme med hvor de ligger deres penge, og derfor ikke overvejer at kigge på nye metoder til at håndtere deres budgetter.
Vi har beskrevet de interne og eksterne forhold ang. budgetmanager, og lavet en SWOT model til at hjælpe med at give et hurtigt overblik.

7.2 Stærke sider:

Vi skal ikke ud og publisher vores app på samme måde som andre udviklere, når vores app er godkendt af Xena vil den være tilgængelig inde på Xena's hjemmeside, hvor deres marketingsafdeling vil lave reklamer på diverse hjemmesider, hvor man som alm. bruger kommer forbi når man browser på nettet.

Budgetmanager er en simpel og optimeret app, som fremstår meget brugervenlig så enhver person der har sat sig en lille smugle ind i budgetter vil kunne bruge appen og let inddatere data.

Du vil som bruger af budgetmanager have mulighed for at sammenligne dit budget med det regnskab du har på Xena, det vil være med til at kunne give et bedre overblik over dine budgetter og se hvor der måske skal optimeres eller hvor der måske er mulighed for at skære lidt ned, på den måde kan du spare penge.

7.3 Svage sider:

Hvis vores budgetmanager ikke bliver reklameret andre steder end Xena, så er det kun folk med kendskab til Xena der vil have mulighed for at finde den og blive kunde.

I budgetmanager vil der i første version ikke være mulighed for at lave balancer for at kunne holde dine aktiver op mod passiver i din virksomhed.

7.4 Trusler:

I dag er det meget let ved hjælp af værktøjer som f.eks. Excel at lave et simpelt budget fremfor at ville ud og investere i et program til det. Det ikke sikkert man som ny iværksætter på markedet tænker det noget man har lyst til at smide penge efter, da det er de færreste iværksættere der har stor egenkapital.

Et krav til at kunne bruge budgetmanager er, at man som bruger bare har lidt kendskab til budgetter, altså at man som min. ved hvad et resultat budget er.

7.5 Muligheder:

Der er stor mulighed for at større virksomheder, der måske ikke har et optimeret system til deres budget/budgetter ville tage brug af en simpel optimeret app som budgetmanager, og på den måde måske kunne holde bedre overblik over deres indtægter kontra deres omkostninger.

Nye kunder af Xena vil kunne købe appen budgetmanager og på den måde få en lettere start i kampen om at opbygge deres budget fra starten, fremfor hvis de stod på bar bund. Det er muligt via Xena at tage et kursus, omkring deres platform og hvordan regnskaber fungere, her kunne vi evt. inddrage vores Budgetmanager.[1]

[1] https://xena.biz/da/support/kursus/

Interne situation	
Stærke sider	**Svage sider**
• Mindre publishering • Brugervenlig app • Godt overblik over budget kontra det hos Xena	• Eventuelt mangel på reklame • Mangel på f.eks. balance
Eksterne situation	
Muligheder	**Trusler**
• Folk der mangler simpelt værktøj til deres budgetter • Xena's kunder • Aftale med revisore	• Værktøjer som "Excel" • For lidt viden om budgetter

8 Metode

Af: Lasse

Vi havde i gruppen en diskussion om hvilket framework der ville passe bedst, til den opgivet projektopgave. Vi har på vores 3. semester arbejdet meget med SCRUM, og det ville være et naturligt valg at bruge dette. Vi har udover SCRUM også kigget på et andet framework, extreme programming, dette fandt vi dog ikke attraktivt, da det store fokus her besidder på pair-programming, og det er et krav i opgaven at alle skal deltage aktivt, og der skal navn på hvad hver person har bidraget med. Ved XP ville der komme to navne på alt hvad vi har lavet, og det vil derfor ikke give et indblik i, hvad hver person har bidraget med til gruppen individuelt. Vi har derfor valgt at benytte SCRUM, da det giver os en masse brugbare værktøjer.

Udover SCRUM vil vi også benytte os af Unified Process metoden, til at hjælpe med at skabe en rød tråd igennem hele arbejdsprocessen. Vi vil også benytte UML notations diagrammer i form af fx Domæne model, SSD, SD, m.v.

8.1 Unified Process og SCRUM

Vi vil som tidligere nævnt benytte SCRUM og Unified Process sammen, for at hjælpe med at danne overblik over hele projektet. Vi vil benytte de fire faser Inception, Eleboration, Construction og transition. Grunden til vi vil benytte både SCRUM og UP er, at vi i vores projekt, har brug for at få en bredere forståelse af domænet, samt hvilke krav der er til løsningen af opgaven, det kan business modeling og requriments faserne hjælpe os med. I SCRUM er det dog også meningen, at man planlægger i starten af projektet, vi vil blot benytte disse redskaber fra UP til og hjælpe os med at danne et samlet overblik, da SCRUM ikke har en decideret metode til dette.

Vi vil benytte os af udtrykket sprints og ikke iterationer. Disse sprints kan hjælpe os med at danne et tidsoverblik, samt give os nogle krav der skal overholdes, i form af fx en definition of done, der skal overholdes før en given tasks kan defineres som færdig lavet.

Derudover har vi valgt sprints, fordi det er er krav at man efter hvert sprint har et ”working increment of software”, som er med til og sikrer at vi hvert sprint får skrevet og publiceret kode.

8.2 Unified Process

8.2.1 Inception

Vi vil starte ud med inception fasen for at få en bedre forståelse af Domænet, samt finde frem til alle de krav der er opsat i opgaven. Vi vil derfor starte med at fokusere på de User stories der har fokus på disse opgaver, og andre relevante opgaver der måtte være, som fx at få skrevet en problemformulering og få opsat nogle mål.

8.2.2 Eleboration, Construction og Transition

Da vi bruger sprints frem for iterationer, giver det god mening at eleboration, construction og transistion er forbundet. Vores User stories bliver lavet på den måde, at analyse og design er forarbejdet på en user story, der skal laves inden implementation kan finde sted. Så i et sprint vil der både være analyse og design samt implementation før en user story, kan blive betragtet som udført. Derudover skal vores definition of done overholdes, som indeholder krav til testning. Deploy af software vil foregå løbende, som en user story bliver færdig, dette gør vi ikke, fordi vi har krav fra en kunde omkring agil udvikling, men for at hjælpe os med at få et overblik over hvor langt vi er kommet i projektet.

8.3 SCRUM

En af de ting der er lagt op til i opgaven og af product owner er, at der er mulighed for at der kan komme ekstra krav til features løbende i projektet. Hvis det skulle være tilfældet, kan vi ligge disse features ind som user stories i vores backlog, og derefter finde frem til, hvilket sprint de skal med i, alt efter hvor stor en prioritet de har. Her giver scrum os en ekstra mulighed for at være agile.

8.3.1 Sprints

SCRUM er som tidligere nævnt delt op i sprints, vores projekt er på 3 uger, og vi har derfor delt det op i tre sprints af en uge af gangen. Vores sprints er tidfast og har en klar definition på hvad der skal være færdig ved udgangen af sprintet. I starten af vores sprints vil vi vælge de opgaver fra vores product backlog, som har højest prioritet for at få lavet projektet, i en rækkefølge der kan være med til at skabe en rød tråd igennem hele projektet.

8.3.2 Roller

Vi har tre roller opfyldt i vores scrum team, vi har en *Product owner* som er Klaus Nørregaard, han har stillet opgaven med de krav der skal opfyldes, og det er ham vi kan gå til ved eventuelle spørgsmål til projektet. Vi har en Scrum master som er Lasse Meldgaard, han står for indkaldelse af møder, samt hjælpe med at udbedre eventuelle impediments der måtte være for Development teamet. Development teamet er den sidste rolle der skal udfyldes, og denne består af alle fire medlemmer i gruppen. Dette team står for at færdiggøre de enkelte sprints, indenfor de fastsatte rammer.

8.3.3 Scrumboard

Vi bruger www.trello.com, til håndtering af vores Scrumboard, hvor vi har listet alle vores User stories op. En af de store fordele ved et scrumboard er at det hjælper os med at holde styr på vores user stories samt task, så vi ved hvad der skal laves i et gældende sprint, og hvem der laver hvad. Det sikrer også at vi hele tiden holder alle i gruppen i gang, da de kan påbegynde en ny opgave, ved blot at se hvilke tasks der er tilbage på boardet.

Vi vil desuden bruge planing poker til at tidsangive vores stories, det gør vi for at få et så præcist estimat som muligt, og så alle kan klare en opgave inden for den planlagte estimering. Vores scrumboard består af 6 kolonner, som er Backlog, To Do, Work in progress, Test af kode, dokumentation korrektur og en Done.

Backloggen indeholder alle de User stories vi har fundet frem til, de ligger i en prioriteret rækkefølge efter product owners ønske. I vores tilfælde skal vi aflevere et færdigt projekt, og ikke med løbende afleveringer efter hvert sprint, til en kunde. Vi har derfor valgt at prioritere dem efter hvordan rapporten skal opbygges, samt hvordan applikationen skal bygges op. Derudover vil vi tage fat i nogle af de store problemstillinger først, fx hvordan vi kan benytte Xenas api. Det er ikke alle vores stories, der tager udgangspunkt i brugeren, da vi har en rapport der skal laves og vi vil gerne have alle vores opgaver på scrumboarded. Vi vil derfor have nogle User stories der tager udgangspunkt i os som udvikler, der sikrer vi får dokumenteret alt i rapporten.
To Do er vores sprint backlog, hvor vi har opdelt vores user stories i tasks, som vi igen har tidsangivet ved hjælp af planning poker. De tasks vi har i denne kolonne skal være færdige efter sprintet er ovre, hvis de ikke er blevet lavet, vil dette blive taget op i vores sprint retrospective møde.
Work in Progress, er de tasks som er under udvikling af en fra teamet, gruppemedlemmerne navne bliver noteret på de task de rykke fra To-Do til WIP, for at holde styr på hvem der er i gang med hvad.
Test af kode, er hvor vi flytter en task hen når den er klar til at blive testet, vi har et krav om der som minimum skal være whitebox testing af alle tasks, før de kan flyttet over i Done. Det samme gør sig gældende på dokumentation, hvor der i kolonnen ”dokumentation korrektur”, hvor der skal læses

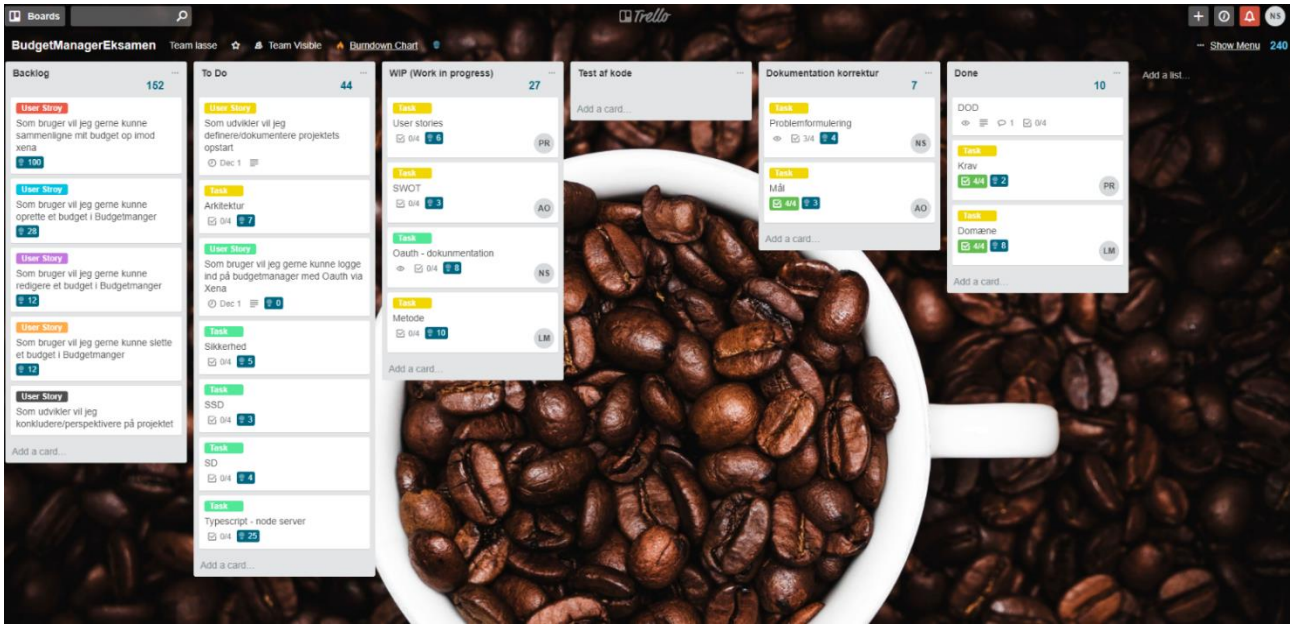
korrektur, på den skrevne dokumentation. På den måde får vi spottet eventuelle fejl og mangler inden, vores arbejde bliver erklæret Done, som er den kolonne det bliver flyttet i når alle krav er overholdt. De krav der skal overholdes som er defineret i vores Definition of done er:

1. Dokumentation er skrevet

2. Kode er testet (White box)

3. Dokumentation læst igennem

4. Alt er i versionskontrol



8.3.4 Møder

Vi kommer i vores team til at holde 3 forskellige slags møder daily scrum meeting, sprint planning meeting og sprint retrospective. Vi holder hver morgen vores daily scrum meeting, det foregår udenfor arbejdsrummet, hvor computere kan være en distraktion. Til mødet er det scrum master og development teamet der deltager, der er sat 10 minutter af til mødet. Der vil hver morgen blive stillet tre spørgsmål til alle deltagere til mødet:

- Hvad lavede du i går?
- Hvad skal du lave i dag?
- Er der nogen hindringer i vejen for arbejdet?

På denne måde får alle et indblik i hvor langt man er kommet i det nuværende sprint. Skulle der være nogle hindringer for arbejdet, skal disse løses.

Vi holder vores sprint planning meeting hver mandag morgen klokken 8.15. Her finder vi de User stories der skal i vores sprint backlog for det kommende sprint. Vi laver ikke et sprint goal, da vores product owner, normalt ikke vil deltage på disse møder. Det sidste møde, sprint retrospective, holder vi hver fredag klokken 13.30, her vil vi igen fokusere på 3 spørgsmål, disse vil blive stille i åbent forum og ikke til hver enkelt deltager.

- Hvad skal vi begynde at gøre?
- Hvad skal vi stoppe med at gøre?
- Hvad skal vi fortsætte med at gøre?

Efter der er kommet ideer/svar til disse spørgsmål, vil vi i samarbejde finde til enighed om, hvilket af disse ting der skal ændres eller bibeholdes. På næste retrospektive vil, der blive kigget på om det er lykkes at ændre på de pågældende ting.

8.4 Agile Manifesto

I og med SCRUM er et agilt framework følger det værdierne fra Agile Manifesto. Det består af 4 sætninger, med 2 påstande i hver. Disse sætninger er med til at definere agil softwareudvikling, og sikrer at man bruger agil softwareudvikling. Sådan som definitioner af sætninger er skrevet, er der lagt op til følgende udsagn: ”Der er værdi i punkterne til højre, men vi værdsætter punkterne til venstre mere”. Dette vil ikke nødvendigvis passe 100 procent til vores projekt, men dette vil vi komme nærmere ind på under hver værdi.
De fire sætninger er følgende:

- **Individer og samarbejde** frem for processor og værktøjer.
- **Velfungerende software** frem for omfattende dokumentation.
- **Samarbejde med kunden** frem for kontraktforhandling.
- **Håndtering af forandringer** frem for fastholdelse af en plan

8.4.1 Individer og samarbejde frem for processor og værktøjer.

SCRUM er et holdbaseret framework, hvor samarbejder er en nødvendighed, for at bruge det effektivt. Der er mange møder, der kræver åbenhed og ærlighed fra alle medlemmerne i grupper, og derfor er vi nødt til at finde et fælles ståsted, der kan fungere for alle i gruppen, for at opnå et godt samarbejde. Vi vil dog stadig benytte processor og værktøjer, til at hjælpe med vores planlægning og udførelse. Et værktøj vi bruger, som også er en del af SCRUM, er vores Scrumboard, det vil vi have et aktivt fokus på at benytte, da det er med til og sikre vi når alle vores opsatte krav.

8.4.2 Velfungerende software frem for omfattende dokumentation

SCRUM handler om af få afleveret et velfungerende stykke software ved hvert sprint. I mange tilfælde vil der, derfor være stor fokus på at få lavet netop dette stykke software færdig, da der er en kunde på den anden side, som forventer at det er klar til aflevering efter sprintet. I vores tilfælde er det ikke sagen, vores produkt, rapport og software, skal afleveres samlet. Det er også en systemudviklingseksamen, vi vil derfor have lige så stor fokus på dokumentationen såvel som en velfungerende software, da vi bliver vurderet på vores rapport.

8.4.3 Samarbejde med kunden frem for kontraktforhandling

Vi har fået listet en række krav til vores projekt, hvilket fungere som vores kontrakt. Der er som tidligere nævnt mulighed for, at der vil komme nogle ekstra opgaver ind over projekt fra product owner. Skulle dette være tilfældet, er der nogle nye krav vi skal tage stilling med i samarbejde med kunden/product owneren, for at komme frem til det bedst mulige resultat, på trods af det ikke var nedskrevet i opgaven fra projektopstarten. Vi er derfor også nødsaget til at samarbejde med kunde frem for at kontraktforhandle.

8.4.4 Håndtering af forandringer frem for fastholdelse af en plan

Dette udsagn passer godt med SCRUM, vi har mulighed for i hver sprint opstart at håndtere og prioritere diverse ændringer, der måtte opstå i vores projekt, i og med vi ikke har planlagt de næste tre uger, men kun en uge frem. Ud fra vores oprindelig tidsangivelse på vores user stories, har vi ekstra tid i projektperioden, hvis der skulle opstå nogle uforudsete problemer, der gør vi bruger længere tid på en task, end først angivet, eller hvis der måtte komme nogle ekstra krav fra product owner.

8.5 Cynefin framework

Vi kan bruge cynefin til at hjælpe med at finde ud af oplysninger omkring vores domæne, samt hvordan dette skal anskues. Om vi har brug for hjælp fra eksperter, eller om opgaverne er nogle vi selv kan håndtere. Dette kan vi gøre efter vi har placeret vores domæne i en af de 5 segmenter i modellen, simple, complicated, complex, chaotic og disorder.

Efter gennemlæsning af det opstillede projekt, havde vi alle en god forståelse af domænet og de krav der er til opgaven. Vi har samlet i gruppen et fælles kendskab til budgetter og regnskaber, da det er noget vi tidligere har arbejdet, med under andre forhold. Derudover var der en god beskrivelse af regnskaber og budgetter i opgaveoplægget. Ud fra disse oplysninger befinder vi os i det simple segment, i det ordnede univers. Her er et ”cause- and-effect” forhold, hvilket passer godt med vores viden omkring domænet, vi kan se hvilke resultater vi får, ud fra de handlinger vi laver, inden vi foretager os dem. Dette gør også at vi

basere de fleste af vores beslutninger på facts, og den viden vi allerede har omkring domænet, eller en viden der vil være os lettilgængelig.

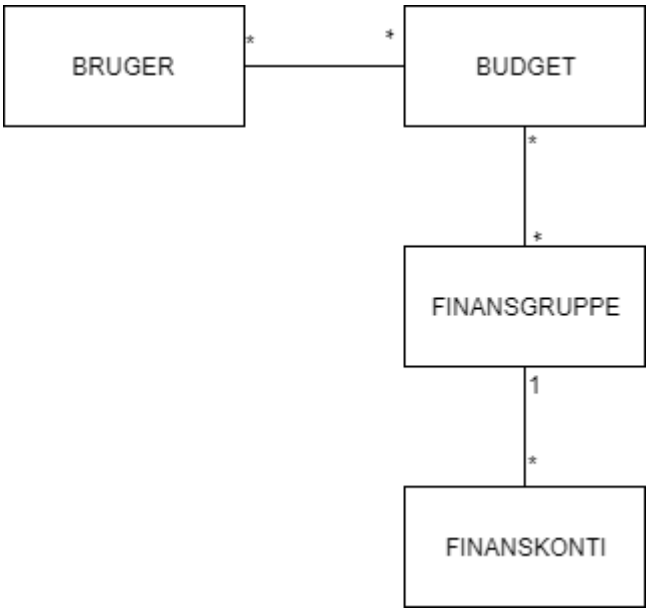
Man kan argumentere for, at vi kunne bevæge os over i det kompliceret segment, hvis vi tog Xena med i vores vurderinger omkring domænet. Vi skal benytte Xenas API'er, og vi har ikke meget viden omkring dette. Her er vi afhængig af den dokumentation de levere omkring, hvilke data vi skal sende med, samt hvilke data, vi kan forvente at modtage. Vi skal her bruge ekstra tid på analyse af hvilke API vi skal bruge, derudover skal vi have oprettet nogle test data, på deres applikation, som vi kan bruge til at teste vores applikation. På dette område kan det også blive aktuelt, at vi møder udfordringer, der kræver at vi tager kontakt til Xena, for at kunne udbedre dem.

9 Domænemodel

Af: Lasse

Efter gennemlæsningen af den udleverede projektopgave, var en af vores første opgaver at få en bedre forståelse af det domæne, som vi skal beskæftige os med. Vi har valgt at fokusere på vores eget domæne Budgetmanager, og ikke Xena, da vores løsning bliver en ekstern applikation udenfor Xena, der skal kunne snakke sammen med Xenas api, og hente data fra et firmas regnskab.
Vi har dog brugt elementer fra Xenas domæne, som finansgrupper og finanskonti, da det er de konceptuelle klasser, vi skal sammenligne data med.

Vi havde en kort snak med vores product owner Klaus, omkring hvad hans forventninger til løsningen indebærer mht. til hvor specifik et budget skulle være i forhold til et regnskab. Vi kom frem til der skulle være mulighed for, at inddatere tal for finanskonti i budgettet. Alternativet ville være kun at sammenligne tallene på finansgrupper, men på denne måde får vi en mere realistisk, komplet og dybdegående sammenligning med Xena regnskaberne.



Vores domæne model består af 4 konceptuelle klasser, Brugere, Budget, Finansgrupper og Finanskonti.

En bruger kan have flere budgetter, da der skal være mulighed for at oprette flere budgetter for fx flere virksomheder, eller budgetter for flere år.

Et budget kan også have flere brugere, da man som bruger i Xena bliver tilføjet til en virksomhed, og da der kan være flere bruge på samme regnskab i Xena, skal dette også være en mulighed for vores løsning.

Et budget kan have flere finansgrupper. Der vil som standard ved oprettelse af et projekt være tilkoblet de nødvendige finansgrupper der skal til, for at oprettet et fyldestgørende budget.

En finansgruppe kan også eksistere på flere budgetter.

En finansgruppe kan have mange finanskonti. Der skal være mulighed for at tilkoble de finanskonti som en virksomhed, måtte finde nødvendige for at lave et budget der passer til deres behov.

En finanskonto kan kun være tilkoble til en finansgruppe, da en postering som fx ”salg af cykler”, kun skal være posteret under omsætning, og ikke samtidig også kunne fremkomme under omkostninger.

De finanskonti der vil være mulighed for at oprette, vil blive hentet fra det respektive regnskab inde fra Xena. På denne måde får vi alle de konti der er relevante for et firmas regnskab med i budgettet. Dette resulterer i at regnskabet skal have oprettet finanskonti i Xena, før det er muligt at hente dem over i Budgetmanager.

10 User stories

Af: Patrick

Projektets opstart

Som udvikler vil jeg definere/dokumentere projektets opstart, så vi kan følge vores arbejdsproces i forløbet for hver uge. Vi vil her definere hvordan vi har tænkt os at starte projektet, sådan at vi efterfølgende får det bedst mulige grundlag for resten af projektet. Vi vil også dokumentere hvordan vi igennem forløbet, har arbejdet med de forskellige dele af projektet.

Login

Som bruger vil jeg gerne kunne logge ind på budgetmanager med Oauth via Xena, så jeg som bruger er sikker på, at det kun er personer med gyldigt login til Xena som kan bruge budgetmanager. Men selvom andre brugere har et gyldigt login til Xena, betyder det ikke at de har mulighed for at kunne benytte en anden brugers budget. De brugerer som har adgang til de oprettede budgetter skal være medlem af den pågældende virksomhed på Xena. Vi har valgt at bruge login da det giver brugeren en sikkerhed, da personer der ikke har adgang heller ikke kommer til at kunne se budgettet i budgetmanageren.

Oprette

Som bruger vil jeg gerne kunne oprette et nyt budget i budgetmanageren. Dette vil give brugeren mulighed for at kunne inddatere, hvordan brugeren forventer sin omsætning og omkostninger vil fordele sig for en given periode. Dette giver brugeren mulighed for nemt at få et overblik over, hvor stor en difference der er på det budgetteret regnskab frem for det reelle. Derfor kan brugeren nemt få et overblik over hvordan en forventet periode vil se ud. Brugeren får ud fra det oprettede budget, et værktøj til hvordan de forventede omsætning - og omkostningstal ser ud.

Redigere

Som bruger vil jeg gerne kunne redigere et budget i budgetmanager, fordi der kan forekomme uforudsete udgifter som har en stor betydning for andre punkter i budgettet. Ved at oprette denne funktionalitet vil dette give brugeren mulighed for at kunne tilføje eller ændre et budget for en given periode, hvis der skulle være sket en tastefejl i navnet eller beløbet. Men brugeren har også mulighed for at kunne tilføje en uforudset udgift, som der kan have en større indflydelse på resten af budgettet.

Slette

Som bruger vil jeg gerne kunne slette et budget i budgetmanager. Som bruger er det en god mulighed at kunne slette et helt budget ad gangen hvis det skulle være nødvendigt.

Inddatere

Som bruger vil jeg kunne se inddaterede data for et specifikt budget, sådan at brugeren vil kunne inddatere i et oprette budget. Dette giver brugeren muligheden for at kunne inddatere indtægter og udgifter for et specifikt budget, sådan at brugen kan inddatere oplysningerne for eks. Januar.

Sammenligne

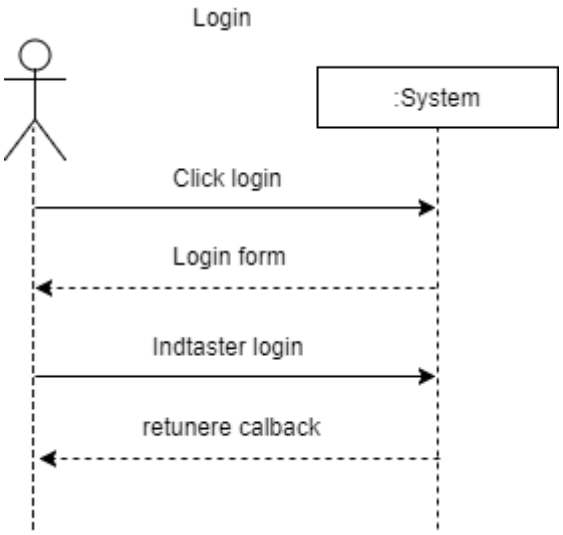
Som bruger vil jeg gerne kunne sammenligne mit budget op imod Xena. På Xena vil et regnskab blive præsenteret for en given periode. Som bruger giver dette mulighed for løbende at kunne inddatere i Xena, omkring hvilke salg og omkostninger man som virksomhed har foretaget sig. Her vil vores bruger kunne sammenligne sit budget med regnskabet fra Xena. Her vil brugeren kunne få et overblik over om det forventede budget er blevet overholdt eller om udgifterne er højere end indtægterne.

Perspektivere

Som udvikler vil jeg konkludere/perspektivere på projektet, som det sidste inden projektet afsluttes. Her tager man som udvikler udgangspunkt i hvordan projektet har forløbet og giver en opsummering af hvad man har undersøgt i forbindelse med projektet. Hertil kommer man også ind på hvordan opgaven er løst i forhold til hvilken teori og metoden man har brugt gennem udviklingen. Dette giver udviklerne for et projekt mulighed for at opsummere perioden, metoder og den teori man har undersøgt og benyttet.

11 System sekvens diagram

Af: Patrick



Vi har valgt at lave en SSD[2] for loginprocessen. Dette diagram skal være med til at give et overblik over, hvilke handlinger der sker mellem systemet og brugeren. Dette sikre at man ikke overser nogen af de handlinger der sker mellem brugeren og systemet.

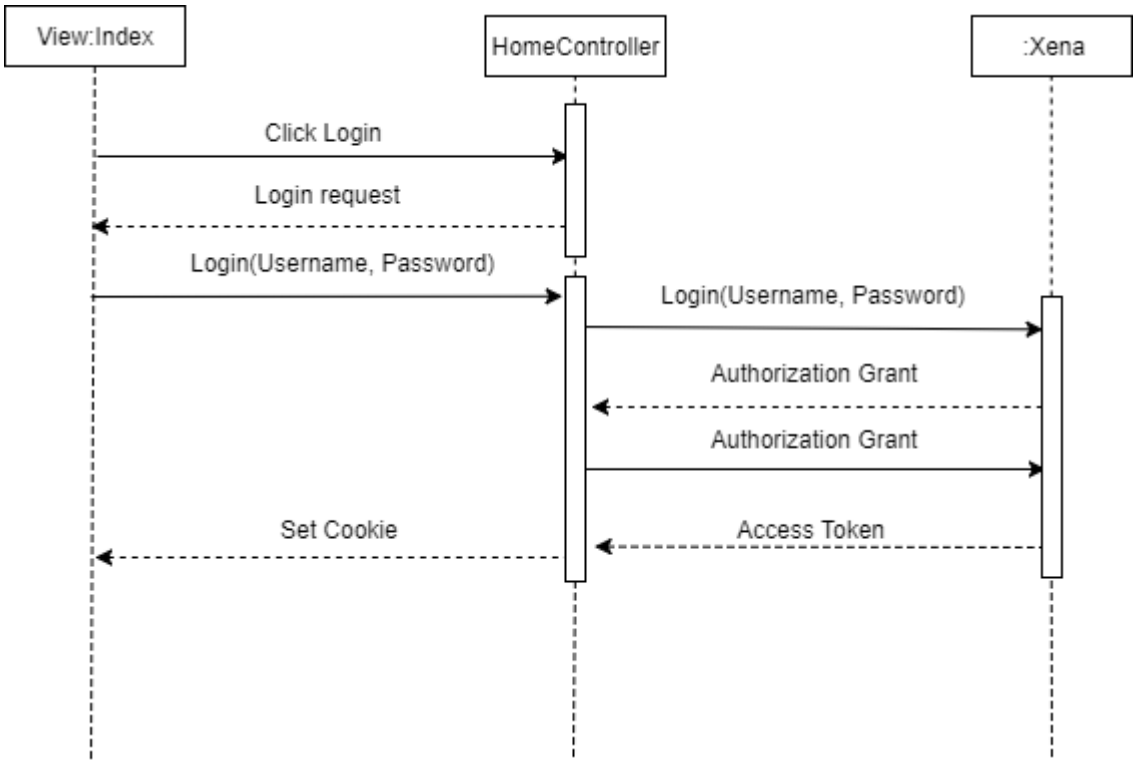
Vores SSD for ”login” viser hvordan processen ser ud, for vores brugers handlinger og hvordan systemet håndtere brugerens handlinger. I dette tilfælde for vores SSD ønsker brugeren at logge ind på vores budgetmanager, dette gøres ved at systemet returnere 2 tekstbokse hvor man skal indtaste brugernavn og adgangskode. Når brugeren har indtastet oplysningerne og trykker login, bliver brugeren af systemet dirigeret ind på en ny side, hvor brugeren nu kan benytte funktionaliteten af budgetmanageren.
De resterende SSD’er kan findes i bilaget.

[2] System sekvens diagram

12 SD for login

Af: Patrick

Vores SD[1] viser interaktionen mellem de forskellige klasser. Dette giver et visuelt billede, af hvilken rækkefølge interaktionerne sker i.

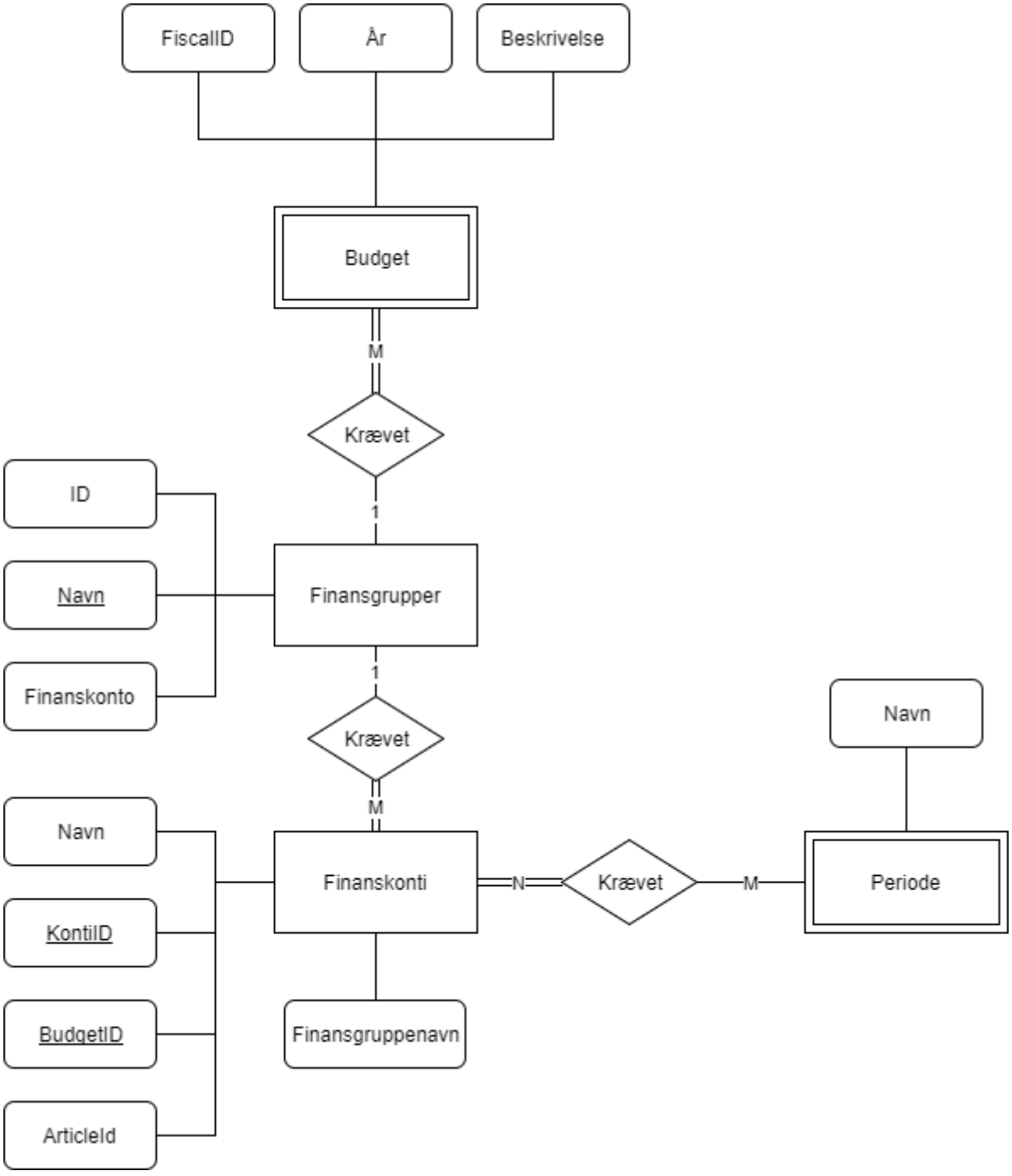


Ud fra vores SD[3] kan vi se at når vores budgetmanager bliver åbnet, bliver brugeren bedt om at logge ind, for at kunne benytte vores budgetmanager. Processen for login bliver brugeren verificeret ved brug af Oauth2, hvor brugeren giver adgang til brugerinformationer fra brugerens Xena konto. På den måde er man helt sikker på at personen der vil logge ind, er ham han udgiver sig for. Når brugeren har indtastet sine loginoplysninger bliver de sendt til Xena, for at validerer om man har en konto. Når brugeren bliver godkendt bliver en 'authorization grant' sendt tilbage til budgetmanageren, som sender den tilbage til Xena. Dette returnere en 'Access token' som så kan bruges fremad rettet i samme session, sådan at man ikke skal logge ind hver gang.

[3] Sekvens diagram

13 ERD

Af: Nikolaj



Efter vi havde analyseret domænet har vi udarbejdet et ERD. På baggrund af de informationer vi har fået fra projektoplægget og PO, har vi fundet frem til dette resultat.

13.1 Budget

Vores Budgetentitet har 3 attributter: FiscalID, År og Beskrivelse.

FiscalID'et er et ID som er givet af Xena når en virksomhed oprettes i deres system.

År er til at definere hvilket år budgettet er for.

Beskrivelse er til eventuelle kommentar omkring budgettet, da der kan være flere brugere der arbejder med det samme budget.

Budget er en svag entitet da der ikke er en åbenlys primær nøgle.

Budget har en relation til Finansgrupper som er krævet, da der ikke kan eksistere et budget uden finansgruppe.

Denne relation er en, én til mange relation. Da et budget kun kan have én finansgruppe, men én finans gruppe kan have mange budgetter.

13.2 Finansgrupper

Finansgruppeentiteten har kun én attribut og dette er Navn. Navn er på samme tid primærnøglen for denne entitet, da vi har besluttet at gruppe navnet er unikt etc. Der kan ikke være 2 finansgrupper der hedder Omsætning.

Finansgrupper har en relation til Finanskonti, denne relation er en, én til mange relation, da én finansgruppe kan have mange finanskonti og én finanskonti kan kun have én finansgruppe.

13.3 Finanskonti

Finanskontientiteten har 2 attributter: Navn og KontiID.

Navn er navnet for finanskontien som bliver fremvist for brugeren på klient siden.

KontiID er et ID som er givet fra Xena ved oprettelse af Finanskontiere, da denne er unik vil den være en perfekt primær nøgle.

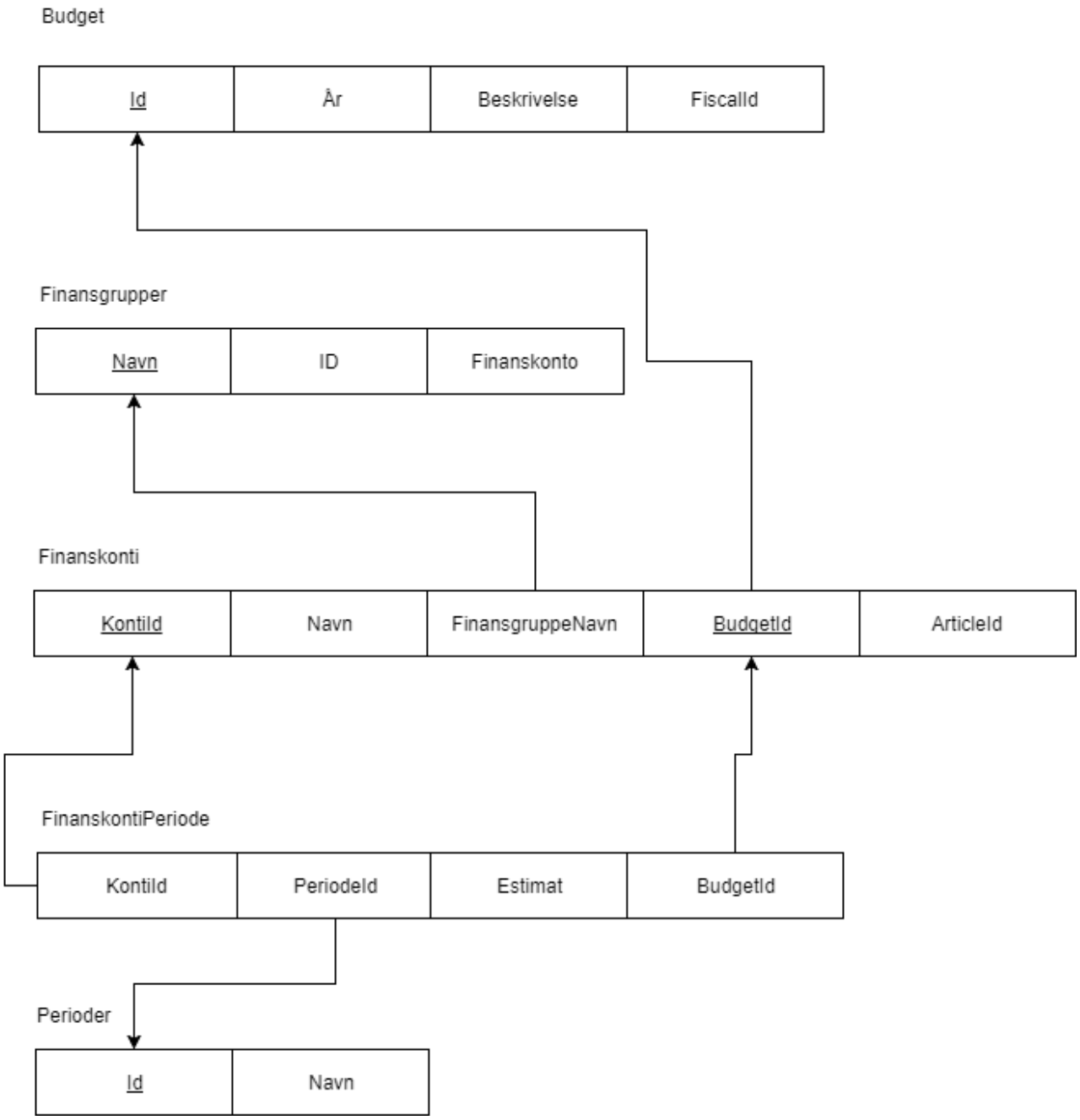
Finanskontientiteten har en mange til mange relation med Periodeentiteten, det er i denne relation vi vil oprette en relations tabel, hvori vi kan inddatere vores estimater. Det er en mange til mange relation, da der fx skal kunne oprettes 12 finanskonti der hedder det samme, men er koblet på hver deres måned.

13.4 Periode

Periodeentiteten har 1 attribut, som er Navn. Det er her vi kan inddatere de måneder, der er til rådighed for budgettet. Dette er en svag entitet da der ikke er nogen åbenlys primærnøgle

14 Mapping

Af: Lasse



Vi har været fælles i vores gruppe omkring, at lave mapping til vores database. Der er mange faktorer vi skulle tage højde for, og målet var at ramme det rigtige resultat første gang, så vi ikke senere skal ind og foretage ændringer i databasen. Grunden til vi vælger at bruge mapping er, at den giver os et overblik over designet, alternativet var at lave vores queries direkte fra vores ERD. Med mapping får vi overblik over alle vores tabeller, primary keys og foreign keys, og hvordan vores queries skal oprettes. Vi kan fx ikke oprette

Finanskonti tabellen, før vi har oprettet Finansgruppe tabellen, da der er en foreign key i Finanskonti som peger på Finansgrupper.

I vores mapping har vi 5 tabeller, hvoraf 1 er en samlingstabel mellem finanskonti og perioder.
Vi har valgt at bruge en surrogatnøgle i Budget tabellen, da vi som tidligere nævnt, ikke har nogle entiteter vi kan bruge som primary key.
Finanskonti har en foreign key fra Navn på finansgrupper, dette er nødvendigt for at få koblet de to tabeller sammen, så vi har mulighed for at se hvilken gruppe en konto hører under. Det samme gør sig gældende for id fra Budget, hvor vi også har en foreign key. Derudover har vi en composite key, der er sat sammen af KontoId og BudgetId. Vi har valgt dette, da KontoId i sig selv ikke er unik, da det er et tal vi henter ud fra Xena, og flere forskellige regnskaber kan indeholde det samme KontoId. På denne måde får vi skabt en unik nøgle der samtidig er koblet på et budget.
Vores Perioder tabel indeholder som vores budget tabel også en surrogatnøgle, da vi ikke har en værdi vi ellers kan bruge som primary key.
Den sidste tabel er vores FinansKontiPeriode, som er en samlingstabel mellem Finanskonti og Perioder. Denne tabel bruger vi til at opbevare vores estimater til budgetterne. Det estimat bliver så koblet sammen med en periode og en finanskonto via 2 foreign keys. Derudover er der en foreign key fra Finanskonti på BudgetId, dette er påkrævet da vi har en composite key i vores finanskonti tabel, og derfor skal vi have begge keys med i samlingstabellen. Vi har været nødt til at lave en samlingstabel, da det skal være muligt, at give estimater på en Finanskonti på flere forskellige perioder, dette kunne være for fx måneder.

14.1 Normaliseringsregler

Da vi lavede vores mapping, har vi samtidig haft fokus på de tre første normaliseringsregler, dette har vi gjort for at gøre databasen mere fleksibel og fjerne overflødige data. Det kunne fx være at have den samme data gemt i forskellige kolonner, ved at følge reglerne har vi kun data gemt et sted, og skal ikke rette data i flere forskellige tabeller.

14.1.1 Første normalform

- Fjern gentagende grupper i individuelle tabeller.
- Opret en særskilt tabel for hver sæt af relateret data.
- Identificer hver sæt af relateret data med en primærnøgle

Vi har opfyldt de tre punkter i den første normalform, vi mener at den data der er relevant for en tabel, er placeret i den pågældende tabel. Vi har ingen gentagende grupper i individuelle tabeller, alt vores data skal kun rettes et sted. Vi har også en primær nøgle til hver sæt af relateret data i de tabeller, hvor det er relevant.

14.1.2 Anden normalform

- Opret særskilte tabeller til værdisæt, som kan anvendes til flere poster.
- Relater disse med en fremmednøgle.

Vi har oprettet vores mapping efter, at der skal være særskilte værdisæt i alle tabeller, som kan anvendes til flere poster. Vi har igen ingen data der er duplikeret og som skal ændres i flere tabeller ved hver rettelse.

14.1.3 Tredje normalform

- Fjern de felter der ikke afhænger af nøglen

Vi har også fjernet de felter, som ikke er nødvendige for definition af en tabel, et eksempel på dette er Finansgruppe navnet hvor vi har oprettet en særskilt tabel, da det skal være muligt at trække en komplet liste over alle finansgrupper, selvom de ikke nødvendigvis er blevet brugt af en finanskonto. Man kunne også argumentere for, at år kolonnen i Budget tabellen, kunne flyttes ud i sin egen tabel. I vores tilfælde mener vi dog, at år er relevant for at definere et budget, og på den baggrund hører til Budget tabellen.

Vi har valgt kun at holde fokus på de første tre normaliseringsregler, da det er i vores design er nok til at have en database, hvor funktionaliteten stadig er intakt.

15 Sprint 1

15.1 Sprint planing

Af: Anders

Vi har i første sprint valgt at tage vores 2 userstories, ”Som udvikler vil jeg definere/dokumentere projektets opstart” og ”Som bruger vil jeg gerne kunne logge ind på budgetmanager med Oauth via Xena”, fra vores backlog. Vi havde estimeret 90 timer til første sprint. I dette sprint var der udelukkende fokus på at vi fik lavet alle vores modeller rigtigt for at undgå konflikter i fremtiden når vi skulle lave vores database og begynde at kode.
Hvis der skulle være blevet underestimeret angående vores tid, så vil der i den resterende tid blive arbejdet på at få udviklet en god database, som skal være fundamentet for at programmet køre ordentligt og vi kan arbejde med de data vi ønsker.

15.2 Sprint 1 Review

Af: Lasse

Vi havde et møde med vores PO, da vi skulle opbygge vores database, for at sikrer den blev oprettet med de tabeller og kolonner, der skulle til for at sikre en gunstig database. Der var et par småting der skulle justeres, men ellers var PO godt tilfreds med det der blev vist frem på mødet. Derudover havde vi sparring med PO omkring vores Oauth.

15.3 Sprint 1 Retrospective

Af: Lasse

På mødet fik hver person i teamet svaret på de tre spørgsmål, der hører sig til et Retrospective møde. Der var nogle små justeringer, vi skulle tage højde for, som fx at være bedre til at finde alle de tasks der måtte være til en user story, da der var et par småting der var lidt uklare. En ting vi var enige om var, at vi skulle blive ved med at være gode til at opdatere Scrum boarded, da det giver os en god fornemmelse af hvor langt vi er i processen.

16 Arkitektur

16.1 Client-Server arkitektur

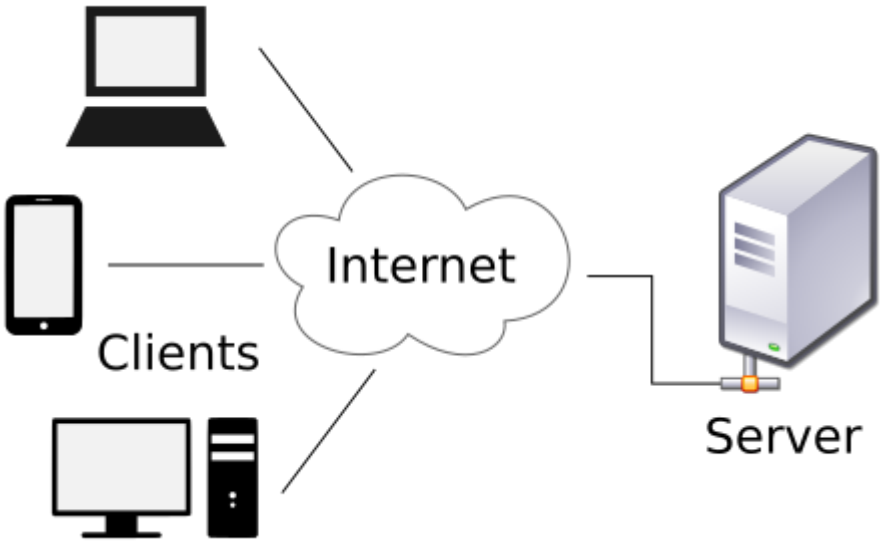
Client-Server er en arkitektur af et computernetværk, hvori mange klienter laver request til en central server(host).

Klient siden indeholder et interface / gui, også kaldet front-end, hvorpå der kan laves request til serveren, og et interface til at fremvise den returnerede data.

Serveren venter på requests fra klienten og svare på dem. Klienten har nødvendigvis ingen kendskab til hvordan serveren fungere, dette kaldes også back-end.

Klienter er for det meste en arbejdsstation eller en personlig computer, hvorimod en server er en kraftigere computer der kan håndtere en masse trafik.

Client-Server ariktekturen er effektiv i situationer hvor klienten og serveren har forskellige arbejdes fordelinger, fx På en hospitals computer, kan klient computeren køre en applikation hvori en bruger kan indtaste patient informationer, og server køre et andet program der står for at inddatere disse informationer i en database[4].



[4] https://www.britannica.com/technology/client-server-architecture

16.1.1 Fordele og Ulemper[5]

16.1.1.1 Fordele

Fordele ved at bruge Client-Server arkitekturen med vores Budgetmanager, er følgende:

- **Data deling**
  Det er muligt for flere klienter at til gå den data der ligger på serveren på samme tid. Der kan tilføjes og læses data på samme tid.
- **Tilgængelighed**
  Der er fri tilgængelighed til Budgetmanager fra hvor som helst, så længe brugeren har internetadgang. Det er ikke krævet at man skal være på et bestemt netværk eller lign.
- **Forskellige systemer**
  Om du er på en mac, pc eller andet. Er det fri mulighed for at tilgå serveren. Du skal blot have internetadgang, så kan man tilgå serveren. Dette gør at man ikke skal tage forbehold for hvilke systemer der skal bruge serveren.
- **Vedligeholdelse**
  Med denne arkitektur kan man nemt opdatere eller lave ændringer til Budgetmanageren uden at det påvirker klienten. Det er dog nødvendigt at have flere servere for at systemet skal fungere hvis man deaktivere én eller flere servere.
- **Sikkerhed**
  Ved at vi bruger Client-Server arkitekturen, sikre vi at serveren har bedre kontrol over dataene, og at det kun er autoriserede bruger der kun har adgang til dataene.

16.1.1.2 Ulemper

Selvom der er mange fordele ved Client-Server arkitekturen, har den også sine ulemper.

- **Server overload**
  Hvis der er meget trafik på netværket, kan serveren blive overloaded. Hvilket betyder at den ikke kan følge med til alle de request der bliver kaldt fra klient siden, og derved får klienten en dårligere/langsommere oplevelse. Som når Skat, udgiver forskudsopgørelsen og hele den danske befolkning på samme tid vil tilgå deres data.
  Vi ser ikke dette som et problem med vores site, da der ikke forventes den mængde trafik.

[5] http://clientserverarch.blogspot.dk/2013/03/advantages-and-disadvantages-of-client.html

- **Downtime**

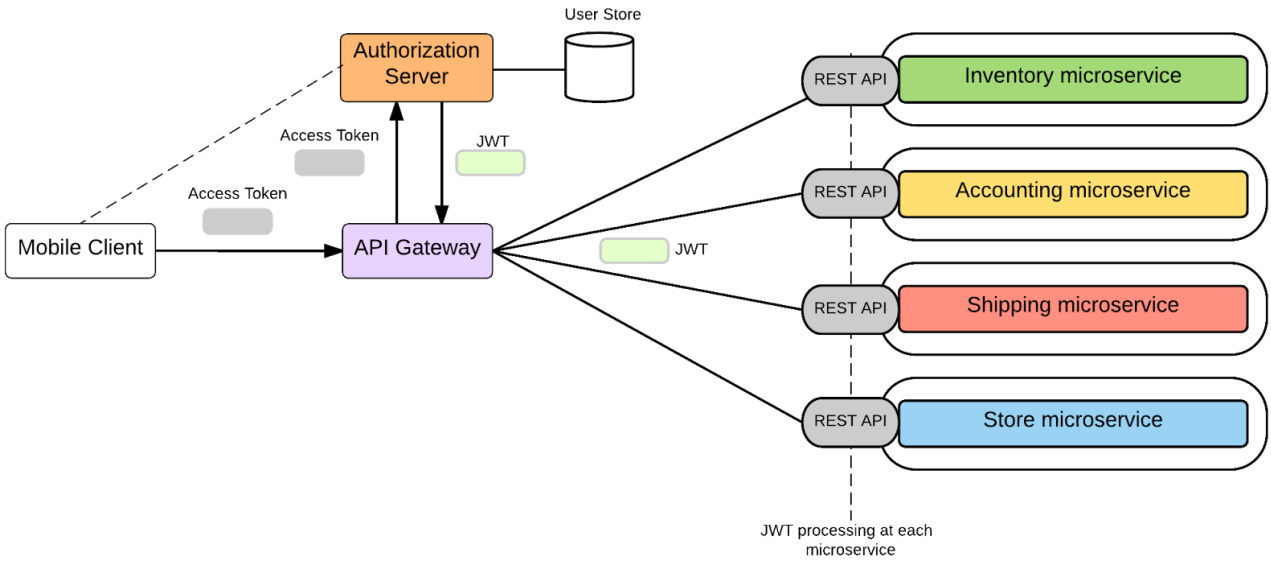
  Hvis en vigtig server brænder sammen eller lign. Vil klientens request ikke blive behandlet og systemet går offline. (Dette kan løses ved at have backup servere)

16.2 Microservice arkitektur

Af: Anders

16.2.1 Hvad er microservices

Microservice er en software arkitektur som består af en eller flere små applikationer som for det meste vil være implementeret i skyen som eksempelvis en virtuel maskine. Ved at de køre på en virtuel maskine giver det bedre mulighed for at administrere hvor meget CPU samt hukommelse der bruges af servicen. Microservice arkitektur bruges til at lave enkelte applikationer som passer til andre servicer. Hver microservice vil normalt have deres egen database frem for en stor samlet.
For at der kan kommunikeres mellem de andre services bruges der API'er.



16.2.2 Fordele ved at bruge microservice med budgetmanager

Ved brugen af microservice vil man som udviklere arbejde i mindre fokuseret grupper da der kun skal fokuseres på en service. Det kan også have effekt på koden da der er betydeligt mindre end hvis vi stod med en almindelig service, der er mindre kode at sætte sig ind i når det skal implementeres.

Hvis en service går ned betyder det at det kun er den ene service der ikke virker mere, da en service er uafhængig af alle andre services.

Det giver mulighed for at opdatere en service uden at man er nød til at ligge hele applikationen ned, og hvis der under opdateringen skulle ske mindre fejl, vil det betyde at det stadig kun er denne ene del der ikke er aktiv.

16.2.3 Ulemper ved at bruge microservice med budgetmanager

Det skal sikres at den microservice vi laver ikke kan være skyld i at hele systemet går offline, fordi vores microservice er afhængig af andre services. Hvis f.eks. en anden microservice er blevet opdateret kunne det ske at en bestemt service ikke er kompatibelt mere.

Der vil være risiko for at hver microservice er skrevet i et nyt sprog eller med nyt framework, hvilket kan være med til at gøre det svært at opretholde servicen. Derfor kan det være en fordel at der er sat nogle få regler op som man skal holde sig indenfor, det vil dog være i strid mod at hvert udviklingsteam selv bestemmer deres arbejdsmetode.

16.3 Vores valg

I vores system vil det give god mening at bruger client-server arkitekturen, da klienter skal kunne inddatere deres budgetter og hente informationer ud på samme tid.
Derved har vi en front-end, klient side, og et back-end, server side. Som kan håndtere de nødvendige opgaver, der skal til for at systemet fungere bedst muligt.
Ved at vi laver vores Budgetmanger i MVC, har vi udfyldt client-server arkitekturen, ved at brugeren ser deres ønskede views, og laver nogle requests der bliver behandlet af vores controllers. Dataene vil blive inddateret i en SQL-database ved hjælp af nogle kald fra vores controller, her ved har vi en client-server arkitektur.

Det vil ikke give meget mening at bruger microservices i vores budgetmanager da der ikke er behov for splitte arbejdet ud over flere services og processere.

17 Sikkerhed

17.1 Oauth2

Af: Nikolaj

Oauth er et verificeringsframework, der giver en applikation begrænset adgang til bruger data fra en HTTP service, som fx Facebook, Github osv.
Frameworket virker ved at overføre bruger verifikationen, til den service hvorpå bruger informationerne ligger. Herefter skal brugeren give applikationen tilladelse til at tilgå deres information.
Der er 4 roller i Oauth:

1. Bruger
2. Klienten
3. Autentificering server
4. Ressource server

17.1.1 Bruger

Brugeren også kaldet ressource ejer, er den bruger som giver en applikation tilladelse til at tilgå deres information.

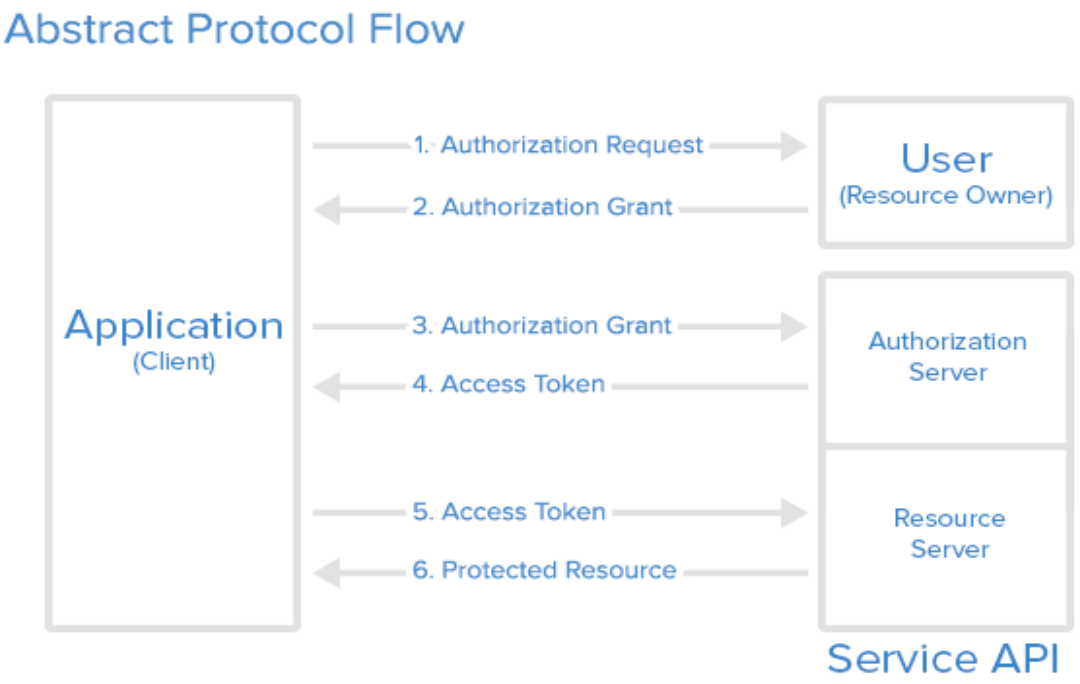
17.1.2 Klienten

Klienten er den applikation der gerne vil tilgå brugerens informationer. Før den kan dette skal den have tilladelse fra brugeren, og valideres af HTTP servicen.

17.1.3 Ressource / Autentificering server

Ressourceserveren er der hvor brugerens information er lageret, og Autentificering serveren verificerer identiteten af brugeren og returnere en access token til klienten.

17.2 Oauth flow



1. Applikationen laver et login request til brugeren
2. Hvis brugeren giver sine logininformationer til applikationen sender han en godkendelse.
3. Applikationen requester en access token fra autentificering serveren, ved at sende sin egen identitet med og godkendelsen fra brugeren.
4. Hvis applikationens identitet verificeres og bruger godkendelsen er valid, sender autentificeringsserveren en access token til applikationen.
5. Applikationen sender et request til ressourceserveren, og vedhæfter sin access token.
6. Hvis det er en valid access token returnere ressourceserveren de informationer applikationen requester.

17.3 Applikation registrering

Før man kan bruge Oauth på sin applikation skal den registreres på den service hvor man vil hente data fra fx Xena.

Servicen skal bruge nogle informationer om applikationen:

- Applikations navn
- Applikations website
- Callback URL

Callback URL er der hvor man bliver overført til når autentificeringen er godkendt.

17.4 Vores valg

Vi har valgt at bruge Oauth på vores applikation, da det giver en høj sikkerhed og giver os adgang til alle de data vi skal bruge fra Xenas ressource server.
Vi har i et tidligere projekt arbejdet direkte med Xena omkring Oauth, og derfor er det et klart valg at vi udnytter vores viden omkring denne verificerings service i dette projekt.

17.4.1 Andre muligheder

Vi havde en anden mulighed for verificering ved Xena. Dette var API-Keys, API-Keys er mindre sikkert end Oauth, da det er krævet at der bliver sendt en key med i hvert request til Xena APIen, Xena og vores applikation kan derved heller ikke være sikker på hvem der sender disse requests, da det kan falsificeres hvis man får fingrene i api-keyen. Hvorimod med Oauth tildeles man en access token, der er unik for den enkelte bruger så man altid kan identificere hvem der laver hvilke requests. Oauth giver også brugeren mere tryghed, da det er noget man kan genkende fra andre applikationer.

17.5 GitHub – versionskontrol

Af: Anders

Vi har valgt på gruppen at tage GitHub i brug som vores versionskontrol værktøj. Det et værktøj vi har arbejdet meget med det seneste semester og er et meget mere fleksibelt værktøj, end hvad vi lærte om på første semester "Team Foundation". Ved brug af TFS[6] lå al vores data på en lokal server, som Klaus havde sat op, som krævede vi tilgik den via. EADanias vpn[7] forbindelse, hvis vi skulle have mulighed for at tilgå den fra andre steder end hvis vi var på samme IP-adresse som serveren. Hvilket som sådan ikke ville være noget problem medmindre nettet som serveren kørte på var nede, risikoen ved at det kunne ske ved en lokal server er større end hvis vi ligger det på en af GitHubs servere.
Derfor er valget taget ud fra vores egen erfaring med begge versionskontrol værktøjer, samt at Github har været en del af vores læringsmål på dette semester at lære om cloud services, som netop github og TFS er. Vi har bl.a. haft besøg af Johan Abildskov, som arbejder i 'Praqma' der tilbyder forskelling kurser, som på klassen lærte os om alt det basiske af hvad github kunne og hvordan vi skulle bruge det.

Det er en mulighed for andre folk at clone hele vores GitHub, dog vil de ikke kunne push til vores GitHub og lave ændringer. Det vil kræve at de bliver tilføjet som collaborators til vores repository. Muligheden for at gøre vores repository privat vil altid være der, hvilket gør at andre folk end os der skal arbejde på det, ikke har nogen form for adgang. På den måde kan vi sikre at vores source code ikke bliver brugt af andre hvis

[6] Team Foundation
[7] Virtuel private network

det, er det vi ønsker. Da det ikke er relevant at skjule vores projekt har vi valgt at lade den være public. GitHub er et meget oplagt valg som vores versionskontrol grundet sikkerheden i at bruge det, samt det er Github vi har størst kendskab til, så vi slet ikke skal sætte os ind i det som vi ville være nød til med TFS. Da det var Klaus der ordnede alt for os ang. TFS vi sku bare bruge den til at dele vores source code.

17.6 Test

Af: Anders

17.6.1 Whitebox

En af vores test består af "whitebox" test. Whitbox testing sker når en person der har kendskab til koden tester den. Det en lidt dyr måde at teste på, da det kræver at en anden sætter sit eget arbejdet på pause for at kunne teste. Fordele ved at lave whitebox test er at der måske vil blive opfanget nogle fejl som udvikleren ikke har opdager, da det en person som forstår koden, men ikke har arbejdet med det specifikke stykke kode.

17.6.2 Blackbox

Ved at køre blackbox test sikre vi at en person uden viden indenfor vores kode tester app'en[8]. Det gør at det ikke er en mulighed at finde eventuelle crash ved hjælp af koden. Vores primære blackbox tester vil være vores PO[9], da han er en erfaren tester, og vi på samme tid kan få respons på mulige ønsker angående tilføjelser til app'en eller hvis der noget vi har misforstået ud fra forklaring.

17.6.3 Destructive

Vores destructive test vil blive udført på samme tid som whitebox og blackbox test, formålet ved denne test form er at prøve alle tænkelig metode at få vores app til at crashe, så vi kan finde ud af hvor der skal sættes tid af for at sikre at når app'en bliver deployed at kunderne ikke vil kunne få programmet til at crashe

17.6.4 Usability

Denne test vil primært blive udført som blackbox test for at vi kan sikre at vores interface er let at forstå for andre end os der har arbejdet på det i en længere periode.

[8] Applikation
[9] Product owner

17.7 Database – valg af data

Af: Anders

Vi har valgt i vores database, at vi ikke vil gemme alle folks oplysninger fra Xena når de vil sammenligne deres budgetmanager med Xena's budget. Dog har vi valgt at det vil være en mulighed at kunne lave en identisk kopi af deres finanskonti navne, id'er og finansgruppe navne, da det vil være nødvendigt at de er helt identiske for at kunne sammenligne de 2 budgetter. Dette vil gøre det lettere for den budgetansvarlige at inddatere al nødvendige data, idet alle finaskontoer, finansgrupper og kontonumre er tilføjet, så der kun mangler diverse omkostninger og indtægter.

18 Sprint 2

18.1 Sprint planing

Af: Anders

Vi valgte i andet sprint at flytte 3 userstories over i vores ”To do” kolonne i scrumboarded. De stories vi flyttede over hed ”Som bruger vil jeg gerne kunne oprette et budget i Budgetmanager”, ”Som bruger vil jeg gerne kunne redigere/inddatere et budget i Budgetmanager”, ”Som bruger vil jeg gerne kunne slettet et budget i Budgetmanager”. Dette kunne vi gå direkte i gang med da vi havde underestimeret vores tid i sprint 1 og derfor nåede at få lavet vores database på sidste dag i sprint 1.

Vi havde to forskellige userstories, der hed redigere og inddatere, disse to har vi valgt at lægge sammen, da inddatere og redigere ville komme til at foregå på sammen måde. Dette sker fordi, at når vi opretter et budget, vil alle finanskonti blive oprette med en værdi på 0, så der altid vil være noget data at sammenligne på. Derfor skal vi ikke bruge en insert funktion i vores db, men blot en update i stedet, og derfor vil vores to user stories blive til 1.
Der vil hovedsageligt blive fokuseret på at skrive kode i dette sprint.
Nogle af vores task i dette sprint er måske blevet over estimeret, da vi før har haft en del problemer med Xena og tilgå deres data, hvis det skulle ske vi blev hurtigere færdig ville vi rykke flere tasks over i To-Do for dette sprint, så vi ikke sad uden nogen opgaver den sidste del af sprintet.

18.2 Sprint 2 Review

Af: Anders

Fredag d. 8/12 havde vi fået booket et møde med vores PO Klaus for at holde ham ”up to date” og sikre at vi havde opfyldt hans ønsker samt have muligheden for at høre ham ad om han havde yderlige ønsker til programmet.

Vi fik vist PO vores produkt som det så ud pt. og der startede en dialog på tværs hvor han spurgte ind til funktionalitet og vi besvarede så godt vi kunne og tog ønsker om eventuelle forbedringer til os. Vi spurgte ind til hvordan PO ønskede oversigten af budgetter samt sammenligningen skulle se ud. Vi mente det var hovedparten af opgaven, så var meget opsat på det blev lavet som han ønskede det.
PO viste på en tavle hvordan han ønskede designet skulle se ud på oversigten af budgettet, her ville han have sat de budgetteret tal op imod de reelle tal fra regnskabet i Xena og under skulle der fremvises differencen i procent.

18.3 Sprint 2 Retrospective

Af: Lasse

Vores andet retrospective møde kom vi frem til at vi skulle blive bedre til ikke at splitte vores task for meget op. Vi havde tasks hvor arbejdet hang sammen, som fx en task der hed ” lave view til opret budget” og en anden der hed ”lave controller til opret”. I vores tilfælde var det den samme person der lavede begge tasks, da controller og view hænger meget sammen i kodningen. Dette kunne have været blevet til en task ”Opret view og controller budget”, på denne måde er der mulighed for både at arbejde i view og controller samtidig. Vi har brugt en del tid på tidsestimeringer af hver task, og det har givet os et godt udbytte i sidste ende, da de timer vi har sat på, i mange tilfælde har været meget nøjagtige, og vi derfor ved præcis hvad vi kan forvente at nå af tasks hver dag. Så dette er en af de ting vi har aftalt at blive ved med.

19 Prototyping[10]

Af: Nikolaj

Prototyping bruges til at repræsentere løsningen af et problem. I vores tilfælde brugte vi det på vores Oauth.

Der er fire typer af prototyping: presentation prototypes, prototype proper, breadboard prototypes and pilot system.

19.1 Presentation prototype

Denne prototype er en, udviklere udleverer til en evt. fremtidige kunde, der skal overbevises om at produktet er værd at investere i.

19.2 Prototype proper

Denne prototype bliver udarbejdet og testet, for at forstå brugernes krav. Imens det reelle produkt stadig er under udvikling.

19.3 Breadboard prototype

Breadboard bruges af udvikler til at teste tekniske opgaver krævet af projektet.

19.4 Pilot system

Pilot prototype er en prototype der indeholder alle de nødvendige aspekter af et produkt, og er tæt på at være det færdige produkt. Man kan kalde denne type prototype for Beta, hvis man skal sammenligne med testning.

19.5 Vores valg

Vi valgte at bruge Breadboard prototyping, da det passer godt ind i vores projekt som udviklere, og vi havde nogle spørgsmål omkring vores Oauth implantation.
Vi lavede en test metode til at hente data ud fra Xena igennem vores Oauth applikation. Dette gjorde vi for at sikre os at vi havde hul igennem. Dette ville spare os en masse tid i sidste ende, da vi ikke skulle bekymre os om at evt. problemer med data ville skyldes vores Oauth.
Protypen er efterfølgende blevet implementeret i der færdige produkt, da den levede op til de krav vi som udviklere havde og som var stillet af projektoplægget.

[10] Prototyping and Software Development Approaches – Mahil Carr & Dr. June Verner

20 Design Studio Method

Af: Patrick

Design studio er en metode hvor hele gruppen samles og laver skitser som skal være med til at give et indblik i hvordan designet på programmet i sidste ende kan komme til at se ud.[11]

20.1 Sketching

Gennem Design Studio skal man i gruppen blive enige om hvilket mål der er ved en opgave, sådan hele gruppen har samme forståelse af opgaven. Efter hele gruppen har brugt tid på at snakke opgaven igennem og dermed har fået en fælles forståelse af opgaven. Skal hvert gruppe medlem lave 6 skitser til produktets design, dette bliver der sat en timer på så man kun har 5 min til at lave 6 skitser. Når hele gruppen har lavet sine skitser, skal hver gruppe medlem præsentere hvad han har tegnet. Når alle i gruppen har præsenteret sine design skal resten af gruppen give konstruktiv kritik omkring designet. Når alle i gruppen har fået kritik til deres skitser, starter man forfra med at skulle tegne skitser.

20.2 Hvordan har vi benyttet sketching

Vi har benyttet sketching til at finde frem til et design af vores interfaces. Her fandt vores gruppe et stille rum hvor vi kunne sidde uforstyrret. Vi fik hver især et papir og et skriveredskab. Vi startede så en timer på 3 min, på denne tid skulle vi så sidde og tegne vores ideer til hvordan et design kunne se ud. Efter tidtagningen satte vi os ned og forklarede hvorfor vi havde tegnet designet som vi havde gjort. Efter vi hver især havde gennemgået vores tegning fandt vi punkter på de forskellige tegninger som vi ikke synes skulle være med i det endelige design. Efter vi var kommet frem til nogle punkter vi skulle have med i designet, indstillede vi timeren igen og begyndte at tegne nye skitser af designet. Denne proces gentog vi lige så mange gange som var nødvendigt til vi kom frem til et design vi synes så ordentlig ud og som ville give brugeren et godt overblik over hvad der skulle udfyldes.

[11] https://articles.uie.com/design_studio_methodology
https://vimeo.com/37861987

21 Designing interfaces

Af: Anders

21.1 Jennifer Tidwell

I vores valg af design har vi taget faktorer, som Jennifer Tidwell beskriver i sin bog ”Designinig interfaces”, i brug til at skabe interfacet for vores ”Budgetmanager”.
Hun fortæller at folk har tildens til at være hurtige og gerne vil frem i en fart, så folk laver en hurtig ”scan” af siden de er på og tager derefter beslutningen om hvilken handling de vil foretage, selvom det måske er den forkerte handling.

21.1.1 Knapper

Vi har valgt at lave få knapper, deriblandt har vi ved hver afslutning en ”prominent done button” som er en knap der uden beskrivelse forklare brugeren at det er her du færdiggøre dit arbejde. Det vises tit med at den er større end andre knapper og nær bunden af siden.

Inde på vores side hvor man skal inddatere data har vi valgt at placere 2 knapper. Den første knap har vi valgt at placere helt i starten for at sikre det er det første som man ligger mærke til. Knappen bruges til at synkronisere grupper og konti fra regnskabet ind i budgettet. På den måde sikres det at konti navnene er identiske, med dem på Xena, og man ikke glemmer at tilføje en, hvis der er kommet en ny konto i regnskabet. Den anden knap vi har er en ”Prominent done button” som i dette tilfælde er til at gemme, den er placeret i bunden. Dette vil sørge for man husker at trykke på gem når alle ens ønskede data er inddateret. Vi har valgt som en ekstra feature at når du klikker væk fra tekstboksen, så er der et ”onclick event” der gør at når man trykker ud af boksen gemmes de data man lige har indtastet, og for at ramme alle brugernes behov beholdte vi gem knappen, for at sikre trygheden om handlingen for brugeren.
Det er muligt at skifte mellem de forskellige måneder ved hjælp af en dropdown boks i toppen af siden. Ved at den er placeret der vil det være første handling der bliver foretaget, og ved klik på en specifik måned vil måneden skifte i toppen af inddateringsformularen.

21.1.2 Escape hatch

I menu baren er der placeret en såkaldt ”Escape hatch” knap. En sådan knap bruger vi for at hjælpe brugeren med let at navigere tilbage til start. Ved at vi bruger denne knap gør at brugeren kan føle en vis tryghed i at hoppe rundt og tjekke forskellige features, samt have muligheden for at kunne starte forfra med et klik. Knappen hjælper samtidigt med at promovere navnet på vores applikation ved at det er en statisk knap som forbliver i menu baren, lige meget hvilken side man er på. Menu baren er statisk og vil derfor forblive det samme, lige meget hvilken siden du befinder dig på. Det hjælper på at brugeren ikke

bliver forvirret og altid have muligheden for let at gå tilbage og starte forfra. Derudover vil menu baren været markeret med en anden baggrund end resten af siden, på den måde undgår vi at den ikke går i et med hele siden, det hjælper med at gøre det simpelt og brugervenligt.

21.1.3 Tekst

Alle finansgrupper er markeret med **fed** skrift for at indikere at en ny gruppe starter, samt baggrunden skifter for hver anden konto for at hjælpe med at skille dem fra hinanden i forhold til hvis hele siden bare var hvid.

21.1.4 Loading indicator

Vi har valgt på vores sammenlignings side at implementere en 'loading indicator'. Da det tager mere end 5 sekunder at loade siden, mener vi det er oplagt med en indikator til brugeren. Ved at vi giver brugeren en indikator at se på, vil det føles som om tiden går hurtigere, og på den måde med til at gøre oplevelsen bedre for vores bruger.

21.2 Gestalt og principper

Gestalt er en design metode der har 6 principper. Disse principper beskriver hvordan man vil se det fulde billede frem for de individuelle dele når man kigger på en gruppe af objekter.
Ved at vælge et princip hjælper det dit design til at blive mere sammenhængende og komplet.

Vi har valgt at bruge 2 af gestalts principper. Vi mener at netop disse 2 kunne være til gavn for valg af vores design og hjælpe brugeren med at kunne finde rundt og ikke bruge for meget tid på at lede rundt i applikationen. De 2 vi har valgt er "Proximity/Grouping" og "Symmetry & Order".

- Proximity/Grouping

  Proximity er når man har flere forskellige elementer omkring det samme emne, som man så gruppere sammen for at gøre det lettere for brugeren at se at de forskellige elementer høre sammen.

Vi har valgt at bruge "Proximity/Grouping" på vores oversigt side.
I budgettet er der lavet bokse til at vise dit reelle budget op mod dit eget budget samt differencen. Disse bokse er lavet som 1 stor boks hvor der er 3 firkanter indeni, de 2 øverste viser posteringer og den nederste viser differencen i procent.

- Symmetry & Order

  Her gælder det om at få designet noget systematisk og simpelt som gør at brugeren ikke ender med at bruge tid på at prøve at finde en sammenhæng

På siderne oversigt, inddatere og redigere har vi taget princippet ”Symmetry & order” i brug, da vi har sørget for at alle vores finansgrupper er sorteret i rigtig rækkefølge i vores sql script ved at give dem et ID. Dette var vigtigt så man som bruger ikke vil komme til at skulle bruge tid på at hoppe frem og tilbage når tal indtastes, da budgetter starter med at vise omsætningen hvor man derefter postere sine udgifter i rigtig rækkefølge. Havde vi ikke gjort dette ville finansgrupperne stå i alfabetisk rækkefølge i stedet og derfor skabe stor forvirring for brugeren.

22 Sprint 3

22.1 Sprint planing

Af: Anders

Mandag morgen den 11/12 blev vores sidste 'sprint planning meeting' holdt. Resten af vores userstories blev smidt over i 'To do' så vores backlog var tom og målet var sat for at vi skulle ramme vores deadline med de forudsætninger vi har. Vores 2 userstories der blev flyttet over var "Som bruger vil jeg gerne kunne sammenligne mit budget op imod Xena" og "Som udvikler vil jeg konkludere/perspektivere på projektet".

I første del af sprint 3 vil der blive fokuseret på at kunne sammenligne budgetter. Når det er ordnet vil resten af sprintet stå på at skulle konkludere/perspektivere hele forløbet og derefter få læst korrektur og sat rapporten rigtigt sammen, så den vil kunne blive afleveret med et funktionelt produkt inden søndag den 17/12.

22.2 Sprint 3 Review

Af: Nikolaj

Vi har i vores sidste og afsluttende sprint, fået styr på de sidste detaljer til vores applikation.
Vi har sikret os at vi har fået feedback på de beslutninger vi har taget, med PO. Vi fik et *thumbs up*, så applikationen er færdiglavet inden for rammerne af projektoplægget.

22.3 Sprint 3 Retrospective

Af: Lasse

Eftersom vi er færdige med projektet, og ikke skal arbejde mere på det, har vi valgt ikke at holde det retrospective møde.

23 Konklusion

Af: Alle

I problemstillingen stillede vi disse spørgsmål:

Hvordan sikre vi at vores brugers data er beskyttet?

For at vi kan sikre at vores brugerdata er beskyttet, har vi valgt at benytte os af Xena's Oauth2 som gør at man som bruger skal bruge sin Xena konto til at logge ind på vores budgetmanager. Når vi sikrer at brugeren skal logge ind via Xena, har vi derfor ikke brug for at gemme nogen følsomme brugerdata i vores budgetmanager eller i en database. Derfor vil vi ikke gemme f.eks. navn, e-mail osv. nogen steder og derfor ikke kunne blive udsat for hacking af brugerdata.

Hvilke udfordringer opstår der ved at hente data ud fra Xenas API?

Vi har før projektet start haft med Xena at gøre, og derfor vidste vi at inden vi gik i gang at deres dokumentation ikke passer på hvilke API kald man skulle bruge. Derfor var det en større opgave at vi skal finde ud af hvilke API kald, vi skal bruge for at få de rigtige værdier som vi skal bruge. Vi har derfor gennem hele projektet skulle prøve os frem og efter nogle forsøg hvor det ikke var lykkes, har vi været nød til at skrive med Thomas fra Xena, som så kunne give os en API vi skulle benytte i stedet for. Vi har derfor også brugt en del tid gennem projektet til at finde de rigtige API'er.

På hvilken måde vil vi sammenligne data fra Xenas budget og de oprettede budgetter i vores Budgetmanager?

Vi har valgt at lave sammenligningen sådan, at i vores oversigt får vi vist vores budgettal og tal fra Xena's regnskab. Vi har også valgt at alle beløb i vores oversigt vises i tusinde, dette vil give brugeren den fordel at man hurtigt kan få overblikket over hvordan budgettet og regnskabet er i forhold til hinanden. Dertil har vi valgt at lave en boks hvor vi viser hvordan budgettet og regnskabet bliver beregnet til en procent visning. Så ud fra procentvisningen vil man også hurtigt kunne skabe et overblik over om det er gået godt eller skidt bare ved at se på et felt.

Der kan opstå nogle udfordringer hvis dataene i de oprettede budgetters finanskonti ikke stemmer overens med de data der er i Xenas, på hvilken måde vil man løse dette?

Vi var i starten omkring hvordan vi skulle håndtere data fra Xena, hvis de ikke ville være magen til vores data i budgetmanageren. Vi har på baggrund af dette valgt at hente alle finansgrupper og finanskonti ud fra Xena. De finansgrupper og finanskonti som Xena har bliver derfor oprettet med det samme brugeren

vælger at oprette et budget. Derfor vil problemet med at der kommer finansgrupper og finanskonti der er forskellige fra Xena's regnskab og budgetmanagerens budget ikke være muligt at kunne forekomme. Så brugeren vil ikke kunne oprette nye finansgrupper og finanskonti til et budget, igennem budgetmanager, men derimod har vi vores synkronisere knap der gør at man kan opdatere, nylig tilføjet grupper og – eller konti, i Xena regnskabet.

Hvordan kan vi give mulighed for at oprette et budget på en intuitiv måde for brugerne?

Vi er kommet frem til at når brugeren skal oprette et budget i vores budgetmanager, skal brugeren kun indtaste navn og år for budgettet. Når dette er indtastet bliver der som nævnt tidligere, bliver der oprettet finansgrupper og finanskontoer i budgettet, som står inde på Xena. Så brugeren ikke skal indtaste eller tage valg omkring hvad et budget skal indeholde, da dette automatisk bliver indsat fra Xena's budgetter.

24 Perspektivering

Af: Lasse

I vores applikation har vi nogle enkelte ting der kunne være lavet anderledes eller som kunne være blevet tilføjet. Vi har dog i stedet haft et større fokus på at få løst de krav der var opsat til opgaven, og ikke på at tilføje ekstra features.

Vi havde også en snak om at have logging med i applikationen, for at vi kunne holde styr på hvem der oprettede budgetter samt foretog ændringer i disse. Grunden til dette var at alle der er tilkoblet en virksomhed, kan foretage ændringer i et eksisterende budget, så logging ville være en god mulighed på dette område, til at se hvem der ændre hvad.

Når vi indlæser en sammenligning mellem budget og regnskaberne, er der en loading indicator, her ville vi også gerne have indsat en annullere knap, da det er en længer varig process, hvor en annullere knap ville give god mening for brugeren, i tilfælde af et fejlklik eller at man fortryder operationen.

Den sidste ting vi havde snakket om, men som også var en stor opgave, var at få diagram sammenligning mellem budgettet og regnskabet. Dette skulle også være med til at give et større overblik over hvordan regnskabet holdet sig op imod budgettet.

25 Bilag

25.1 Litteraturliste

https://articles.uie.com/design_studio_methodology
https://vimeo.com/37861987
https://xena.biz/da/support/kursus/

Prototyping and Software Development Approaches – Mahil Carr & Dr. June Verner

https://www.scrumalliance.org/why-scrum

http://www.creativebloq.com/graphic-design/gestalt-theory-10134960

http://internativa.com.br/mobile/Livro%20-%20Designing%20Interfaces,%202nd%20Edition,%202010.pdf

25.2 Dagbog

27-11-2017: Første dag

Vi har oprettet gruppen Lasse, Anders, Patrick og Nikolaj.
Vi har lavet en gruppe kontrakt, valgt metode og fået oprettet vores user stories. Vi har lavet ´planning poker´ hvor i vi har sat nogle estimater for vores tasks.
Vi besluttede os for at bruge Scrum med 3 sprints, en uge til hvert sprint.

28-11-2017: Anden dag

Vi startede dagen ud med at vi holdte vores daglige scrum meeting fra 08:15 til vi var færdige. Der blev fortalt hvad der var blevet lavet den foregående dag og hvad planen for i dag var.
Efter vores møde gik vi ind i det lokale vi havde fået skaffet tidligt på dagen, vi gik i gang med de tasks vi havde snakket om vi ville tage. Vi fik snakket med vores product owner Klaus i løbet af dagen for at få bekræftet vores tanker.

29-11-2017: Tredje dag

I dag havde vi ikke mulighed for at få et lokale grundet eksamner, vi valgte i stedet at sætte os i vores undervisningslokale og gjorde vores for at arbejdsmiljøet var acceptabelt for os og de andre som sad derinde.
Da klokken blev 08:15 startede vores scrum meeting igen, og da det var færdigt gik vi alle i gang med vores tasks igen.

30-11-2017: Fjerde dag

I dag klokken 8:15 startede vores daglige scrum meeting igen, og fadt ud af hvordan det gik i går og hvad vi skulle lave resten af dagen.

I dag havde vi som i går heller ikke mulighed for at få et lokale på grund af det var reserveret til 5. semester, så vi satte os ind i vores undervisningslokale og gjorde vores for at arbejdsmiljøet var acceptabelt for os og andre i lokalet.

01-12-2017: Femte dag

Vi holdte scrum igen som holdes hver dag. Vi fik rettet vores erd samt skrevet til den og sat den i done sammen med mapping. Databasen blev lavet samt scriptet blev gemt.

04-12-2017: Sjette dag

Vi holdte et nyt sprint meeting for at finde ud af hvilke userstories der som min skulle laves den kommende uge. Efter det holdte vi et hurtigt standing meeting og forklarede hvad der sidst var arbejdet på, hvordan det var gået og hvad man ville i dag.

Der blev startet på kode (primært db kald, controllers og views).

05-12-2017: Syvende dag

Vi startede ud med at læse projektoplægget igen for at sikre vi ikke havde misset noget. Vi har erfaring fra tidligere projekter med at læse oplægget flere gange i løbet af opgaveperioden. Da man løbende vil opdage eventuelle mangler.

06-12-2017: Ottende dag

Fik kodet inddateringsdelen færdig i budgetmanager. Og kigget lidt på css generelt, så der blev lavet et lækkert design.

07-12-2017: Niende dag

I dag blev der arbejdet med at få skrevet en oversigt ud så det stod rigtigt og med de rigtige data.

08-12-2017: Tiende dag

Vi holdte i dag et møde med vores 'Product Owner' Klaus, hvor vi fik vist vores produkt, hvordan det ser ud, hvad vi tænker og hvad han tænkte. Der blev skrevet noter ned om hans ønsker og efter mødet blev der arbejdet for at få gjort dette sprints userstories puttet over i 'DONE'.

Vi sluttede med at alle userstories var opnået og endnu et vellykket sprint var overstået.

11-12-2017: Ellevte dag

Vi startede på 3 og sidste sprint i dette projekt. Rapporten blev sat sammen så den bare manglede korrektur og at blive sat pænt op. Kodemæssigt blev sammenligningen gjort færdig og der mangler diverse udregninger, oprydning osv.

12-12-2017: Tolvte dag

Vi fortsatte i dag med udregningerne af sammenligningen af vores budget i budgetmanageren og regnskabet fra Xena ser ud. Vi har også lavet små rettelser til rapporten og skrev nye punkter.

13-12-2017: Trettende dag

I dag har vi brugt hele dagen på at rette rapporten igennem da alle gruppemedlemmer har læst rapporten igennem til i dag. Så vi gik i gang med at gå hele rapporten igennem og lavede rettelser der hvor vi mente det skulle rettes.

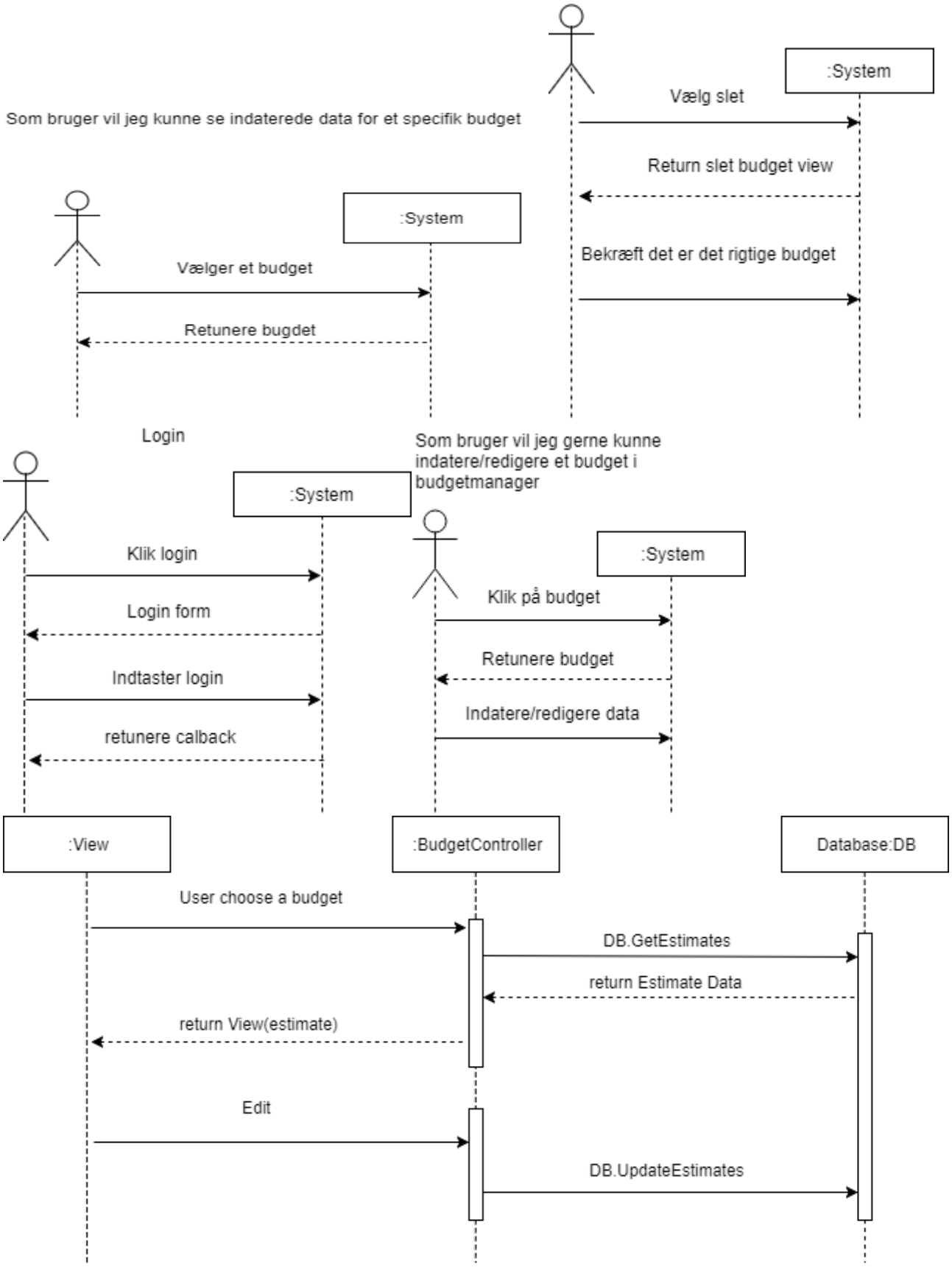
14-12-2017: Fjortende dag

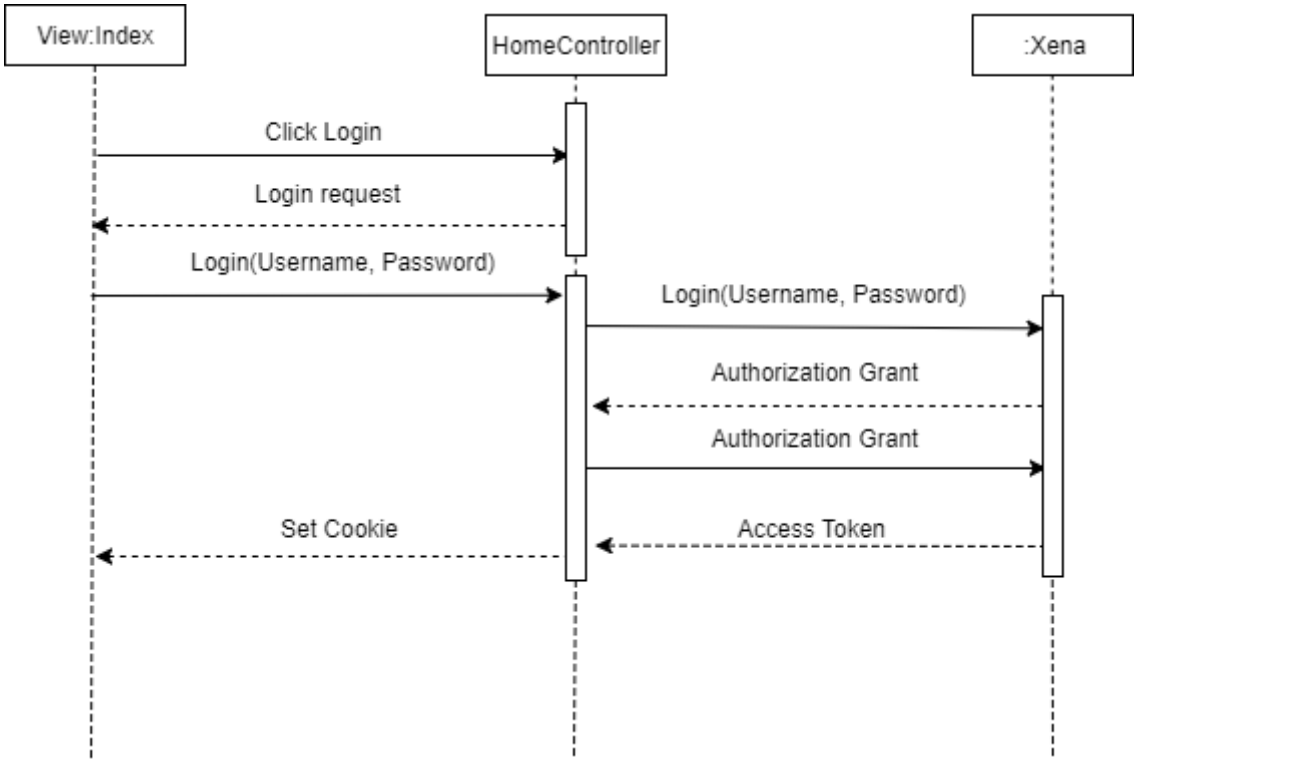
I dag har vi skrevet og omskrevet de områder vi fandt i går. Her kom der nogle nye

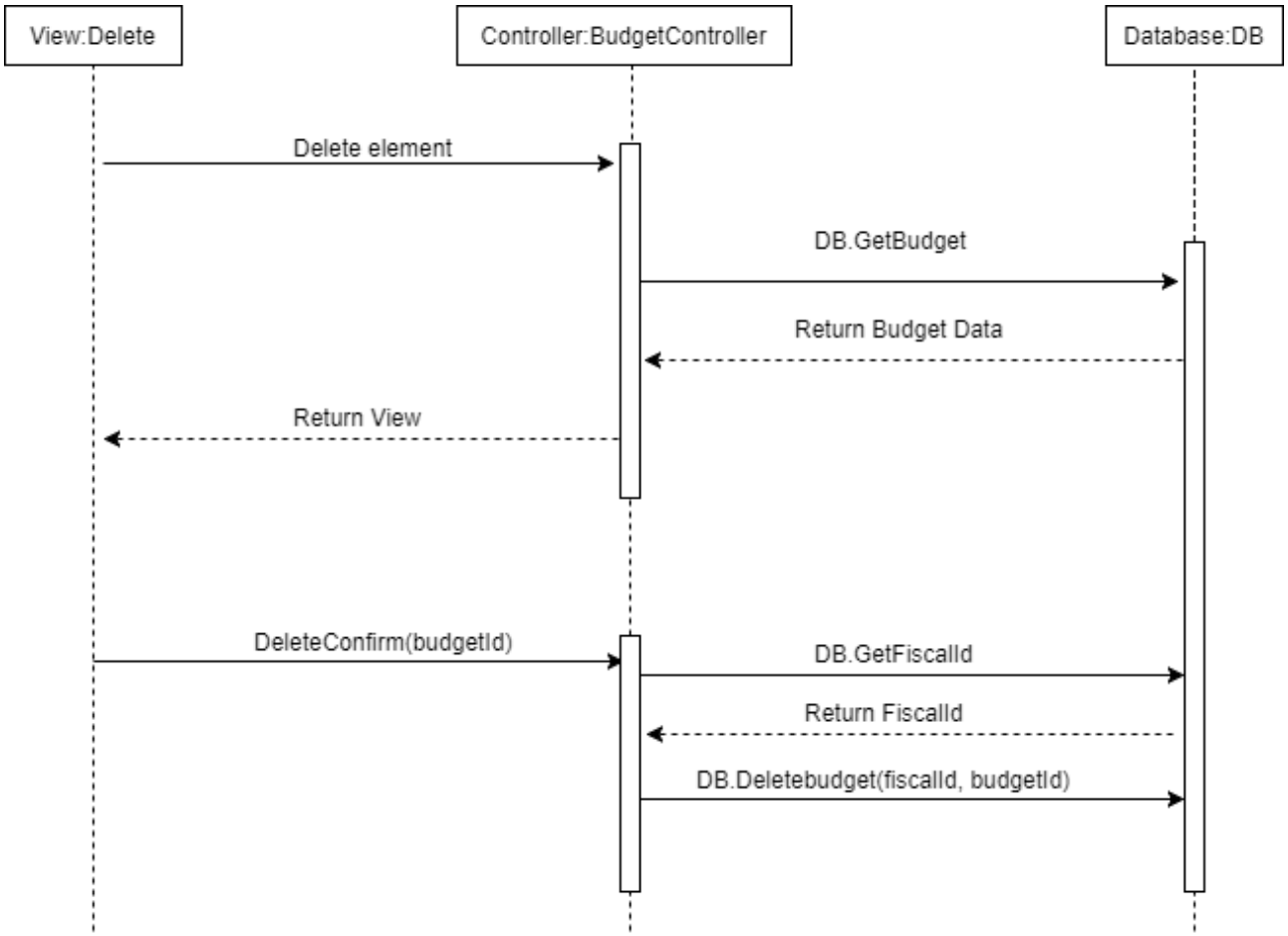
15-12-2017: Femtende dag

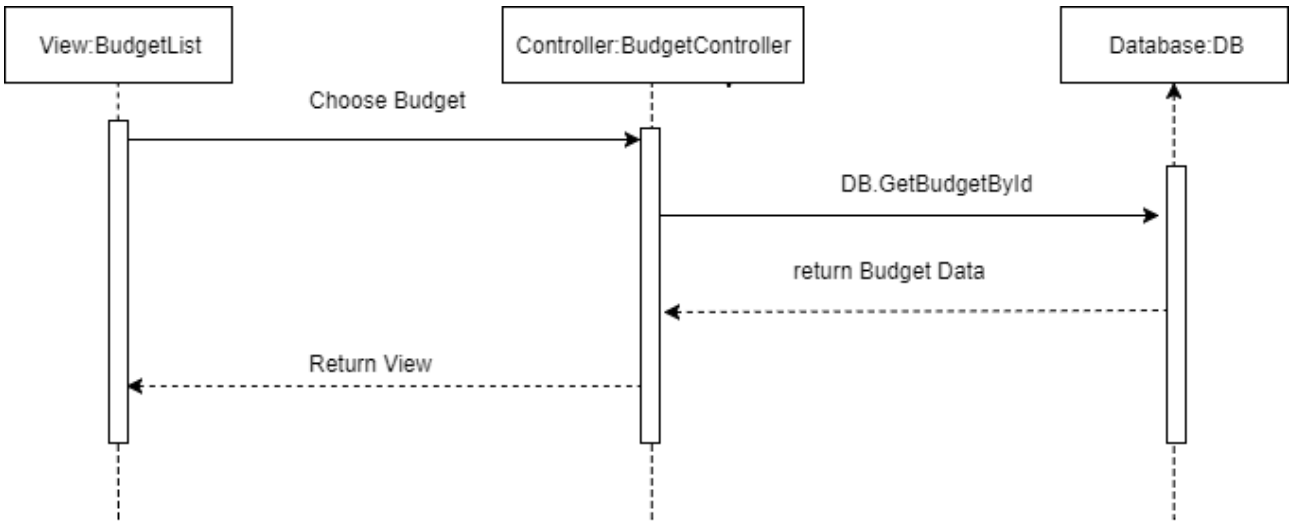
Gennemlæse rapport og lave endelige rettelser. Og opsat rapporten til aflevering.

25.3 Modeller




View:Index
HomeController
:Xena
Click Login
Login request
Login(Username, Password)
Login(Username, Password)
Authorization Grant
Authorization Grant
Set Cookie
Access Token


View:Delete
Controller:BudgetController
Database:DB
Delete element
DB.GetBudget
Return Budget Data
Return View
DeleteConfirm(budgetId)
DB.GetFiscalId
Return FiscalId
DB.Deletebudget(fiscalId, budgetId)


View:BudgetList
Controller:BudgetController
Database:DB
Choose Budget
DB.GetBudgetById
return Budget Data
Return View