1. Indholdsfortegnelse

[2. Projekt opstart 3](#_Toc499802068)

[3. Sprint 1 4](#_Toc499802069)

[3.1. Mål 4](#_Toc499802070)

[3.2. Problemstilling 4](#_Toc499802071)

[3.2.1. Perspektivering af problemstilling 5](#_Toc499802072)

[3.3. Problemformulering 5](#_Toc499802073)

[3.4. Krav til løsning 5](#_Toc499802074)

[3.5. SWOT 6](#_Toc499802075)

[3.5.1. Stærke sider: 6](#_Toc499802076)

[3.5.2. Svage sider: 6](#_Toc499802077)

[3.5.3. Trusler: 6](#_Toc499802078)

[3.5.4. Muligheder: 6](#_Toc499802079)

[3.6. SWOT modellen 7](#_Toc499802080)

[3.7. Metode 8](#_Toc499802081)

[3.8. Domænemodel 8](#_Toc499802082)

[3.9. User stories 9](#_Toc499802083)

[3.9.1 Login 9](#_Toc499802084)

[3.9.2. Oprette 9](#_Toc499802085)

[3.9.3. Redigere 9](#_Toc499802086)

[3.9.4. Slette 10](#_Toc499802087)

[3.9.5. Sammenligne 10](#_Toc499802088)

[3.9.6. Projektets opstart 10](#_Toc499802089)

[3.9.7. Perspektivere 10](#_Toc499802090)

[3.10. SD for login 11](#_Toc499802091)

[3.11. SSD for login 11](#_Toc499802092)

[3.12 Sikkerhed 12](#_Toc499802093)

[3.12.1. Oauth2 12](#_Toc499802094)

[3.12.1.1. Bruger 12](#_Toc499802095)

[3.12.1.2. Klienten 12](#_Toc499802096)

[3.12.1.3. Ressource / Autentificering server 12](#_Toc499802097)

[3.12.2. Oauth flow 13](#_Toc499802098)

[3.12.3. Applikation registrering 13](#_Toc499802099)

[3.12.4. Vores valg 14](#_Toc499802100)

[3.12.4.1. Andre muligheder 14](#_Toc499802101)

[3.12.5. GitHub – versionskontrol 14](#_Toc499802102)

[3.12.6. Test 15](#_Toc499802103)

[3.12.6.1. Whitebox 15](#_Toc499802104)

[3.12.6.2. Blackbox 15](#_Toc499802105)

[3.12.6.3. Destructive 15](#_Toc499802106)

[3.12.6.4. Usability 16](#_Toc499802107)

[3.12.7. Database – valg af data 16](#_Toc499802108)

# 2. Projekt opstart

Af: Anders

Vi startede vores projektperiode mandag den 27-11 klokken 09 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 hele opgaveformuleringen igennem en ekstra gang. Vi satte os hver især ned og skrev hvad vi mente der var vigtigt at fokusere på, derefter diskutere vi hvad alle i gruppen havde skrevet ned.

Vi prøvede at finde et lokale, som vi kunne have for os selv med et stille miljø. Dette var desværre ikke muligt på dag 1. 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 ’ping pong’ mellem hinanden, uden at forstyrre andre eller blive forstyrret. 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 i det endelige resultat af modellen.

Da vi havde et endeligt resultat 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 på hver task og se om det gik op i hvor meget vi havde estimeret til vores user stories.

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 mødte alle op på dag 2. hvor vi havde fået et stille lokale for os selv, hvor der var projektor og tavle som vi kunne bruge til at lave skitser på samt hvis vi havde noget der skulle diskuteres på gruppen så ville det være optimale værktøjer at have hver dag. Vi startede dagen ud med vores ’standing scrum meeting’ hvor vi fik forklaret for hinanden hvad vi lavede den foregående dag og hvad planen var for den pågående dag. Da mødet var slut satte vi os igen i vores lokale hvor rapportskrivningen begyndte.

Vi sørger for at inddrage vores PO når vi mener der er noget vigtigt at diskutere med ham om produktet så vi undgår, at vi pludselig kunne blive stoppet i processen fordi der har været misforståelser fra nogens side om hvad der menes.

# 3. Sprint 1

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. SKRIVES VIDERE SLUT FREDAG

## 3.1. 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 budget man som kunde har liggende på Xena.

Det skal være muligt for kunden selv at kunne skabe nye ”finanskontoer” så der vil være mulighed for at skabe de kontoer der er nødvendig for den enkelte bruger. Samt slette eller redigere diverse kontoer hvis nødvendigt.

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

Vi har valgt at der bruges Oauth via Xena for at man skal kunne logge på budgetmanager, det har vi valgt for at sikre sikkerhed i vores budgetmanager.

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

## 3.2. 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.

### 3.2.1. 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.

## 3.3. Problemformulering

Af: Nikolaj

* 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?

## 3.4. 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
* Overvåge vareforbrug for en periode
* Se hvordan aktuelle konti forholder sig til de budgetterede konti

Ud fra kundens krav til budgetmanageren, er det for kunden at kunne koble budgetter sammen med regnskaber på Xena. Dette giver mulighed for at kunne se uforudsete udgifter og hvilket vareforbrug man har haft i en given periode.

## 3.5. SWOT

Af: Anders

### 3.5.1. 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 deres hjemmeside, hvor andre vil lave mange forskellige reklamer på diverse hjemmeside, 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 budget du har på Xena, det vil være med til at kunne give et bedre og større overblik over ens budgetter og se hvor der måske skal optimeres eller hvor der måske er mulighed for at skære lidt ned og på den måde spare penge.

### 3.5.2. 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.

### 3.5.3. 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.

### 3.5.4. 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.

En mulighed for fremtiden kunne være at få optimeret appen nok til at kunne lave aftaler med revisorer som så kunne tage det videre til deres kunder og forklare om det når de alligevel skal snakke om deres årsregnskab.

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 vil være muligt via Xena at tage et kursus på 2 timer om deres system og budgetter, som kan være med til at fremme forståelsen af vores app på samme tid.[[1]](#footnote-1)

## 3.6. SWOT modellen

Nedenunder er vist en model af SWOT som er med til at give et kort og simpelt overblik over budgetmanagers interne samt eksterne situation.

Interne situationer er handlinger vi selv kan påvirke, hvor eksterne er omverdenen.

SWOT giver et overblik over produktet på et givent tidspunkt, og er med til at evaluere og drage konklusioner, som kan bruges som grundlag for fremtidige handlinger.

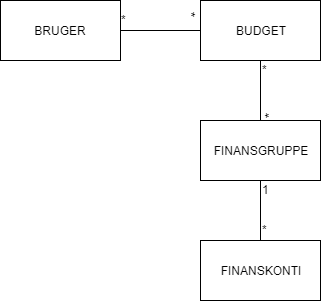
|  |  |
| --- | --- |
| **Interne situation** | |
| **Stærke sider** | **Svage sider** |
| * Mindre publishering * Brugervenlig app * Godt overblik over budget  kontra det hos Xena | * Eventuelt mangel på reklame * Mangel på f.eks. balance |
| **Eksterne situation** | |
| **Muligheder** | **Trusler**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

## 3.7. Metode

## 3.8. 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 både finansgrupperne og 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, 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.

## 3.9. User stories

Af: Patrick

### 3.9.1 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. Brugeren som har oprettet budgettet skal stadig give tilladelse til hvem der har adgang til det pågældende budget gennem budgetmanageren. Hvorfor have login? Dette giver brugeren en sikkerhed, fordi personer der ikke har adgang heller ikke kommer til at kunne se budgettet i budgetmanageren.

### 3.9.2. 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 indkomst og udgifter vil fordele sig for en given periode. Dette giver brugeren mulighed for nemt at få et overblik over, hvor mange penge man har tilbage efter alle faste udgifter er betalt. 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 det forventes at udgifter og indkomster passer sammen.

### 3.9.3. 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. Hvorfor bygge dette? Dette vil 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. På den måde kan han nå at tilpasse nogle andre udgifter til, sådan at der for en periode ikke går i minus i forhold til indtægter og udgifter.

### 3.9.4. 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. Som bruger vil jeg bruge dette til at kunne slette et gammelt budget, hvis denne ikke længere er gældende eller der er blevet oprettet et nyt budget som erstatter den gamle.

### 3.9.5. Sammenligne

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

### 3.9.6. Projektets opstart

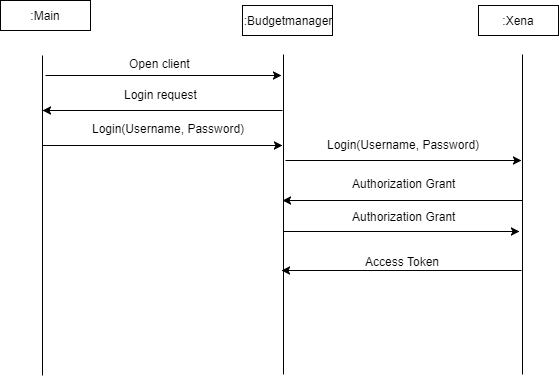
Som udvikler vil jeg definere/dokumentere projektets opstart, så vi kan følge vores arbejdes proces i forløbet for hver uge. Vi vil her definere hvordan vi har tænkt os at starte projektet, sådan vi efterfølgende får det bedst mulige grundlag for resten af projektet. Vi vil også dokumentere hvordan vi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

### 3.9.7. 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.

## 3.10. SD for login

Vores SD[[2]](#footnote-2) viser interaktionen mellem de forskellige klasser. Dette giver et visuelt billede, af hvilken rækkefølge interaktionerne sker i. Interaktionerne bliver arrangeret i den rækkefølge som de bliver sendt i.



Ud fra vores SD 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 til Xena bliver de sendt til Xena, for at blive valideret for om man har en konto. Når brugeren er 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.

## 3.11. SSD for login

## 3.12 Sikkerhed

Af: Anders og Nikolaj

### 3.12.1. 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

#### 3.12.1.1. Bruger

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

#### 3.12.1.2. 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.

#### 3.12.1.3. Ressource / Autentificering server

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

### 3.12.2. 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.

### 3.12.3. 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.

### 3.12.4. 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.

#### 3.12.4.1. 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.

### 3.12.5. GitHub – versionskontrol

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 TF[[3]](#footnote-3) lå al vores data på en lokal server som krævede vi tilgik den via. en vpn[[4]](#footnote-4) 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 IP’en 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 serveren. Få eksempler på at TF ikke vil være tilgængeligt kunne være pga. internetudbyder fejl, vejarbejde der rammer et kabel eller noget helt tredje.

Derfor har vi valgt denne gang at bruge GitHub da omverdenen på samme måde ikke ville kunne påvirke os. Går nettet ned hos udbyderen vil det altid være muligt at koble sig på via. fx mobilt netværk, hvor vi så kan pull seneste ændringer og så arbejder videre med seneste version af projekt og rapport. Det gir os også fordelen at vores data altid vil være let tilgængeligt lige meget om vi er hjemme, på studiet eller et helt andet sted.

Det er dog en mulighed for andre folk at clone hele vores GitHub, dog vil de ikke kunne push til vores GitHub. Det vil kræve at de bliver tilføjet som collaborators til vores repository. Muligheden for at gøre vores repository private 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 soruce code ikke bliver brugt af andre. GitHub er et meget oplagt valg som vores versionskontrol grundet sikkerheden i at bruge det og fleksibiliteten i at vi ikke er bundet af lokale server.

### 3.12.6. Test

#### 3.12.6.1. Whitebox

En af vores test består af ”whitebox” test. Ved at køre whitebox test giver vi source code med til personen som skal teste softwaren. 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 ”dead code” eller andre fejl, da det en person som forstår koden, men ikke har arbejdet med det specifikke stykke kode.

#### 3.12.6.2. Blackbox

Ved at køre blackbox test sikre vi at en person uden viden indenfor vores kode tester app’en[[5]](#footnote-5). Det gør at det ikke er en mulighed at finde eventuelle crash ved hjælp af koden. Vores primære blackbox tester vil være vores PO[[6]](#footnote-6), 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.

#### 3.12.6.3. 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

#### 3.12.6.4. 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.

### 3.12.7. Database – valg af data

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 finanskonto navne, finansgruppe navne og kontonumre, 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.   
Da det ikke er personfølsomme data måtte vi i teorien godt gemme alt deres data så vi kunne implementere det hele i deres nye budget som de kunne arbejde videre i. Men vi mener selv på gruppen det ikke vil være en nødvendighed. På den måde sikre vi også de ikke bare har 2 ens kopier af et budget, men at de får 2 forskellige budgetter de kan arbejde ud fra, og på den måde kan de få hjælp til at finde ud af hvor de skal optimere eller skære ned.

Ved at vi vælge ikke at gemme kontaktoplysninger eller lignende, sikre vi også at vi på ingen måde vil kunne miste oplysninger vi mener andre folk ikke bør få fat i, i tilfælde af hacking.

1. <https://xena.biz/da/support/kursus/> [↑](#footnote-ref-1)
2. sekvens diagram [↑](#footnote-ref-2)
3. Team Foundation [↑](#footnote-ref-3)
4. Virtuel private network [↑](#footnote-ref-4)
5. Applikation [↑](#footnote-ref-5)
6. Product owner [↑](#footnote-ref-6)