

ProjDat2015 - Bibliotek udlånsystem

Oliver Sejling Kogut 010694
Thomas Nyegaard-Signori 141093
Casper Helms 260294

Instruktor: Markus Wittorf

Maj 13, 2015

Indholdsfortegnelse

1	Abstract	4
2	IT-projektets formål og rammer	5
2.1	FACTOR	5
3	Kravspecifikation for IT-løsningen	6
3.1	Funktionelle- og ikke-funktionelle krav	6
3.2	Use case model over system-funktionaliteten	6
3.3	Use cases	7
3.3.1	Lån af bog	8
3.3.2	Udsending af rykker	9
3.3.3	Tilføjelse af bog	10
3.4	Problemområde	10
3.5	BCE-Model	11
3.6	Sekvens diagrammer	12
3.6.1	Lån af bog	12
3.6.2	Tilføjelse af bog	13
3.6.3	Udsending af rykker	14
4	Systemdesign sammenfatning	15
4.1	Kort om systemet	15
4.2	Deploymentdiagram	15
4.3	Datalagring	15
4.4	Adgangskontrol	16
4.4.1	Adgang til systemet	16
4.4.2	Brugertyper og kontrol	16
5	Program- og systemtest	17
5.1	Funktionelle tests	17
5.2	Test af brugervenlighed	17
6	Brugergrænseflade og interaktionsdesign	18
6.1	Lån bog	18
6.2	Rykkere	19
6.3	Præsentation af kørende prototype	19
7	Projektsamarbejdet	20
7.1	I gruppen	20
7.1.1	Hvad går godt	20
7.1.2	Hvad går skidt	20
7.2	Kunden	20
8	Bibliografi	21
9	Litteratur Review	22
9.1	The M.A.D. Experience	22
9.2	No Silver Bullet - Essence and accident in software engineering	23
10	Bilag 1: Versionsstyring	24

11 Bilag 2: Changelog for projektrapporten	26
12 Bilag 3: Timeline	26

1 Abstract

This system is being developed for a library, who now handle their books, loans and customers manually. Our goal is to optimize and simplify both the book and loaner registration, and the actual loans. Besides that, it will be automatizing the creation of reminders, and send them to the loaners by email if necessary. The system will rely on a database where all necessary data are stored, and the actual program will be a website, where you can interact with the system. The system can also work as a self-service system, so the loaner may register a loan or return a book, without talking to an employee at the library.

2 IT-projektets formål og rammer

Vores IT-projekts formål og rammer er beskrevet ved FACTOR-begrebet[s. 39-41, kilde 2]. FACTOR er et værktøj man kan bruge, til at beskrive/definere et IT-projekt kort og præcist.

2.1 FACTOR

Functionality:

Systemet har funktioner, således at man kan registrere brugere, bøger, udlån og udsende rykkere.

Application Domain:

Systemet administreres af ansatte på biblioteket, men kan ved udlån og returnering anvendes af brugerne selv.

Conditions:

Systemet vil blive udviklet i samarbejde med, og efter kundens ønsker.

Technology:

Systemet vil blive udviklet på henholdsvis mac og linux computere. Det bliver udformet som en hjemmeside, som bliver kodet i HTML, CSS og PHP.

Objects:

Bøger og brugere er hovedobjekterne i dette system.

Responsibility:

Systemet skal have ansvaret for at registrere og overskueliggøre brugere, bogbeholdning og udlån.

Det kan læses ud af ovenstående FACTOR beskrivelse af systemet, at formålet med projektet er at udvikle et system, som kan holde styr på brugere, bogbeholdning og de udlån der måtte være. Et andet formål med systemet, er at aflaste de ansatte eller med andre ord, simplificere arbejdet for dem.

Rammerne for projektet er i bund og grund ret fleksible. Vores kunde har naturligvis en række minimumskrav, som de gerne vil have opfyldt. Når dette er gjort vil der være mulighed for videreudvikling eller at foretage ændringer på det allerede udviklede system, hvis der er behov for dette.

3 Kravspecifikation for IT-løsningen

3.1 Funktionelle- og ikke-funktionelle krav

Funktionelle krav:

Funktionelle krav fortæller noget om hvordan systemet fungerer i forbindelse med brugere af systemet og andre eksterne systemer.[s. 119, kilde 1].

- Bibliotekarere skal yderligere kunne registrere henholdsvis bøger og lånere i systemet.
- Systemet skal kunne tjekke for og udsende rykkere/påmindelser på email, til lånere der har overskredet returneringsdatoen.

Når minimumskravene er opfyldt, vil det være muligt for kunden at ønske mere funktionalitet, hvis tiden er til det og det ønskede krav ikke er for stort. Ovenstående er blot tænkt som værende det simpleste endelige system.

Ikke-funktionelle krav:

Ikke-funktionelle krav, er krav der ikke er direkte relateret til funktionaliteten af systemet. Det kan være mange ting, så som systemets ydelse, hvilket programmeringssprog det skal laves i, sikkerhed og lignende.[s. 120, kilde 1].

- Det skal være en hjemmeside, som let kan tilgås fra alle styresystemer via en browser.
- Hjemmesiden skal kun kunne tilgås, hvis man er koblet på bibliotekets netværk.
- Systemet skal generelt sikres mod misbrug udefra, hvad end det er rettet mod bruger oplysninger eller ved input af skadelig tekst i indtastningsfelter.
- Systemet skal være brugervenligt og simpelt, således at det også kan bruges som selvbetjening. Dette vil vi gøre ved at udføre brugertest, og kvalitativ forståelse ved interview med brugere.

3.2 Use case model over system-funktionaliteten

En use case model bruges til at overskueliggøre et system, og bliver i dette tilfælde brugt til at illustrere hvilke funktionaliteter de forskellige aktører har i systemet.[s. 42, kilde 1]

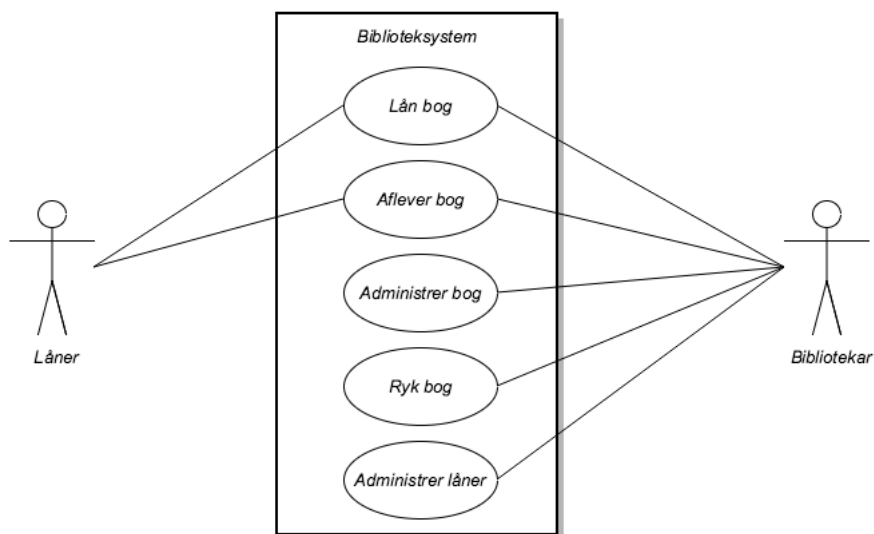


Fig. 3.1: Figuren herover er en illustration af hvilke muligheder, henholdsvis låner og bibliotekar har i systemet.

Figur 3.1 herover skal ikke forstås således, at de to aktører interagerer med hinanden, men blot at de har adgang til forskellige funktioner i samme system. I figuren kan man ligeledes se, at en almindelig låner har mulighed for at låne en bog, eller aflevere en bog igennem IT-systemet. Hvorimod en bibliotekar(eller anden ansat), har samme muligheder, men kan også administrere bøger og brugere, som indebærer at tilføje og slette til/fra databasen, og de har mulighed for at rykke en bog. Med rykke bog menes der en påmindelse, der sendes som mail til en låner, hvis vedkommende har overskredet returneringsdatoen. Hvordan systemet skal kende forskel på disse, bliver formegentlig ved brug af adgangskode eller anden form for identificering af de ansatte, der måtte bruge systemet.

3.3 Use cases

Use cases er en måde at simplificere en del af systemets funktionalitet set fra en aktørs/brugers perspektiv. Altså en beskrivelse af i skridt hvad aktøren skal foretage sig, i hvilken rækkefølge og hvad det resultere i.[s. 42, kilde 1]. Herunder er specificeret tre use cases, som har en vigtig rolle i vores system . Det er henholdsvis lån af bog, udsending af rykkere, og tilføjelse af bøger til systemet.

3.3.1 Lån af bog

<i>Use case name</i>	<u>LånAfBogNy</u>
<i>Participating actor instances</i>	<u>bruger: Låner</u>
<i>Flow of events</i>	<ol style="list-style-type: none"> 1. Brugeren finder en bog han vil låne og går hen til selvbetjenings computeren med den. 2. Brugeren vælger funktionen lån bog. 3. Brugeren indtaster sin mail(hvis han er registreret), og nummeret på bogen. 4. Brugeren har nu registreret lånet, og forlader biblioteket.
<i>Entry condition:</i>	Brugeren skal være registreret
<i>Exit condition:</i>	Brugeren får en bekræftelse, og bogen er udlånt
<i>Quality requirements:</i>	Bogen kan ikke udlånes, før den er blevet returneret

Fig. 3.2: Figuren herover viser, de skridt der skal tages for at låne en bog.

I figur 3.2 herover kan det ses, at ved brug af systemet kræver det kun låneren selv for at kunne låne en bog. Derudover forhindres menneskelige fejl, da en bog kun kan lånes, hvis den er registreret i systemet. Så lånet accepteres ikke, hvis der tastes forkert bognummer. Derudover kræves det er brugeren er registreret i systemet, før denne kan foretage et lån.

3.3.2 Udsending af rykker

<i>Scenario name</i>	<u>RvkBogNy</u>
<i>Participating actor instances</i>	<u>bruger: Bibliotekar</u>
<i>Flow of events</i>	<ol style="list-style-type: none"> 1. På sin computer vælger brugeren funktionen der tjekker for udlån der har overskredet returneringsdatoen. 2. Brugeren får en liste over alle de udlån der har overskredet returneringsdatoen. 3. Brugeren markerer de rykkere hun vil sende, og klikker på knappen send. 4. Systemet sender automatisk en rykker ud på email, til hver af de mails der er registreret på udlån af bøger der endnu ikke er returneret.
<i>Entry condition:</i>	Brugeren skal være logget ind som bibliotekar
<i>Exit condition:</i>	Bibliotekaren får bekræftelse, og rykkerne er blevet sendt
<i>Quality requirements:</i>	Mailen forventes at være ved låneren indenfor 30 sekunder

Fig. 3.3: Figuren herover viser, de skridt bibliotekaren skal tage for at udsende rykkere.

Ved et kig på figur 3.3 kan det ses, hvorledes automatiseringen af henholdsvis tjek og udsending af rykkere fungerer. Det er en klar forbedring af den tidligere metode, hvor de ansatte selv manuelt skulle søge en række bøger igennem, for derefter at ringe lånerne op for at rykke dem. Det kræves naturligvis at det er en bibliotekar der benytter systemet, for at denne funktion er til rådighed.

3.3.3 Tilføjelse af bog

<i>Use case name</i>	<u>TilføjBog</u>
<i>Participating actor instances</i>	<u>bruger: Bibliotekar</u>
<i>Flow of events</i>	1. Brugeren klikker på funktionen tilføj bog. 2. Brugeren indtaster oplysninger på bogen (nr. titel, forfatter). 3. Brugeren bekræfter de indtastede oplysninger. 4. Systemet fortæller brugeren, om bogen er blevet succesfuldt registreret.
<i>Entry condition:</i>	Brugeren skal være logget ind som bibliotekar
<i>Exit condition:</i>	Bibliotekaren får bekræftelse, og bogen er tilføjet
<i>Quality requirements:</i>	Bogen kan udlånes lige efter den er blevet tilføjet

Fig. 3.4: Figuren herover viser, de skridt bibliotekaren skal tage for at tilføje en bog til systemet.

På figur 3.4 herover kan man se hvorledes en bibliotekar tilføjer en bog til systemets database. Hvilke oplysninger på bogen der bør kunne indtastes, er endnu ikke fastlagt, så ovenstående er derfor blot et eksempel. Det kræves igen naturligvis at det er en bibliotekar der bruger systemet.

3.4 Problemområde

Et problemområde er det som systemet skal effektivisere, muliggøre eller løse på en eller anden måde. I vores tilfælde omhandler det altså de bøger, brugere, udlån og rykkere der skal administreres, registreres og lagres i databasen. Alle de dele som systemet er bygget op omkring.[s. 39, kilde 1] Et klassediagram kan bruges til at visualisere problemområdet, på en sådan måde at man kan forbinde de forskellige dele af systemet med hinanden, så man kan se hvordan det er bygget op.[s. 48, kilde 1]

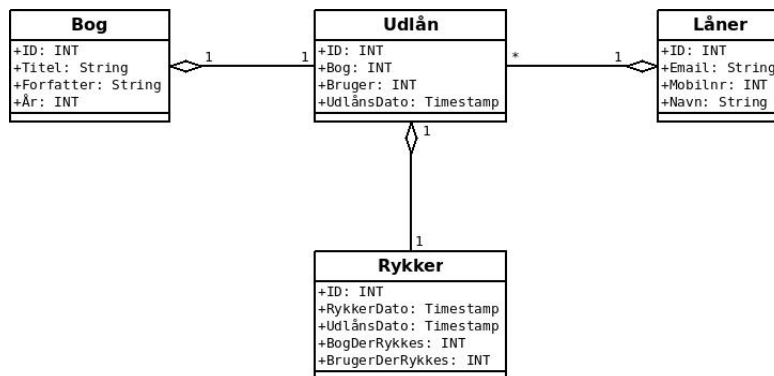


Fig. 3.5: Figuren herover er et klassesdiagram over de klasser, som vores problemområde, og vores system er bygget op omkring.

De givne attributter i objekterne i figur 3.5 herover, er ikke fastlagt, men vi forventer, at det bliver noget lignende det, der er anført. Figuren hænger sammen således, at et udlån består af en bog og en låner, og en rykker består af et udlån(hvori bog og låner er implicit). Alle multipliciteter er 1 til 1, udover at en låner godt kan have flere udlån.

3.5 BCE-Model

Boundary-, Control- og Entity-objekter er begreber, der bliver brugt til at beskrive sammenhængen mellem forskellige dele af et it-system. Ofte forbindelserne mellem brugergrænseflade og de bagvedliggende funktioner og data.

Boundary-objekter er de dele af systemet som en aktør direkte interagerer med. En brugergrænseflade bliver ofte forbundet med en boundary, da dette er hvad en bruger benytter sig af ved brug af et program.

Entity-objekter er de dele af systemet, som det er bygget op omkring. Det er ofte dele af systemet som gemmes på den ene eller anden måde. Det kan ofte forbindes med de entiteter der også normalt findes i en database.

Control-objekter er ikke en decideret del af systemet, men mere et abstrakt begreb for hvad der sker, fra aktøren udfører en handling i et boundary-objekt, til at der igen er noget aktøren skal foretage sig. Man kan forklare det ved, at en aktør eksempelvis klikker på et menupunkt(boundary), herefter sørger et control-objekt for, at der sker det der nu skal ske, som nok ville være at bringe aktøren hen til et andet boundary-objekt, som svarer til det der blev klikket på.[s. 171, kilde 1]

Boundary-objekter:

Brugergrænsefladen i sig selv, altså alle sider og undersider der er i systemet, og de formularer der skal udfyldes ved henholdsvis tilføjelse af bruger og bog, lån af bog osv. Her fremgår en liste over dem alle:

Sider: Forside, Aflever bog, Lån bog, Administrer bog, Administrer bruger, Rykker.

Formularer: Aflever bog form, Lån bog form, Administrer bog form, Administrer bruger form, Rykker form.

Entity-objekter:

Entity-objekter vil være:

Bøger, brugere, udlån og rykkere, i form af data, som hele systemet afhænger af.

Control-objekter:

Control-objekterne er de abstrakte objekter, som sørger for at man bliver ledt de rigtige steder hen, ved valg i menuen. Og derudover også ved afsending af formularer, og derved også kontakt mellem websiden og databasen. Her fremgår en liste over dem alle:

Menupunkternes controllere: Forside controller, Aflever bog controller, Lån bog controller, Administrer bog controller, Administrer bruger controller, Rykker

controller.

Formularenes controllere: Aflever bog form controller, Lån bog form controller, Administrer bog form controller, Administrer bruger form controller, Rykker form controller.

3.6 Sekvens diagrammer

Et sekvensdiagram er en nærmere beskrivelse af en use case, som illustrerer hvordan de forskellige dele af systemet arbejder sammen med hinanden og aktøren, og hvilken rækkefølge det foregår i. Det giver en fornemmelse af hvad en bruger af systemet aktivt skal gøre, og hvad systemet selv sørger for.

3.6.1 Lån af bog

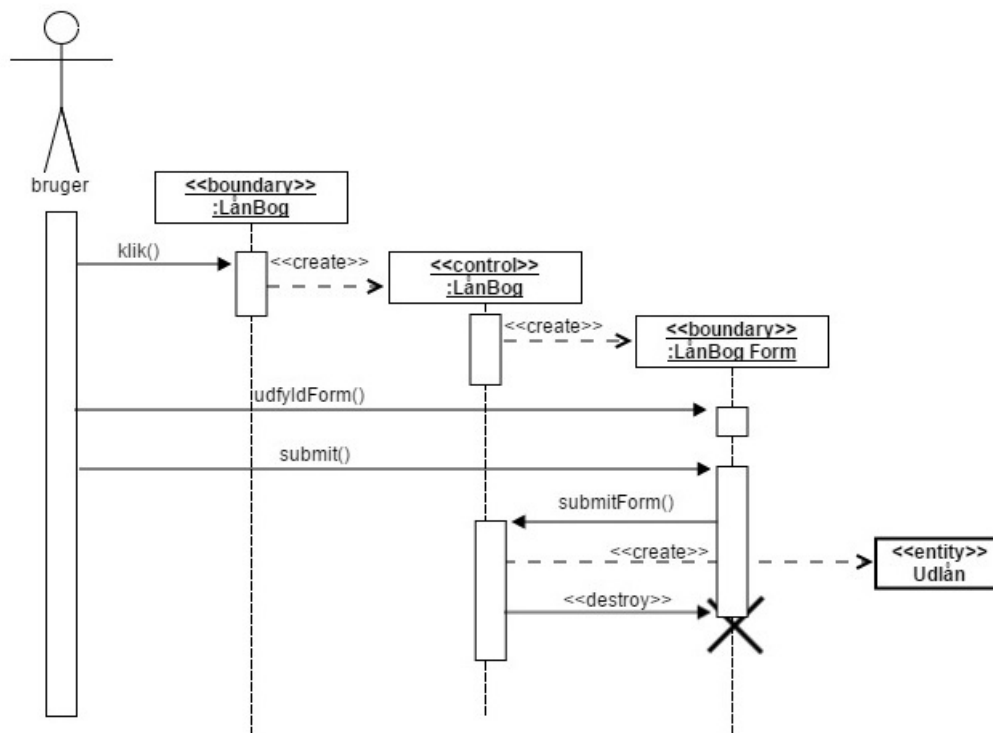


Fig. 3.6: Figuren herover et et sekvensdiagram over use casen der omhandler lån af bog.

I figur 3.6 herover, er sekvens processen delt op i tre dele, menupunktet lån bog, en controller, og form der skal udfyldes. Brugeren skal indtaste sin mail (brugeren skal være registreret for at kunne låne!) og nummer på bogen han vil låne. Herefter klikkes på en knap for at fuldende lånet. Inputtet sendes her til databasen, og den returnerer om hvor vidt lånet blev registreret (hvis mail eller bog ikke findes, vil den fejle. Dette er dog ikke inkluderet i diagrammet herover).

3.6.2 Tilføjelse af bog

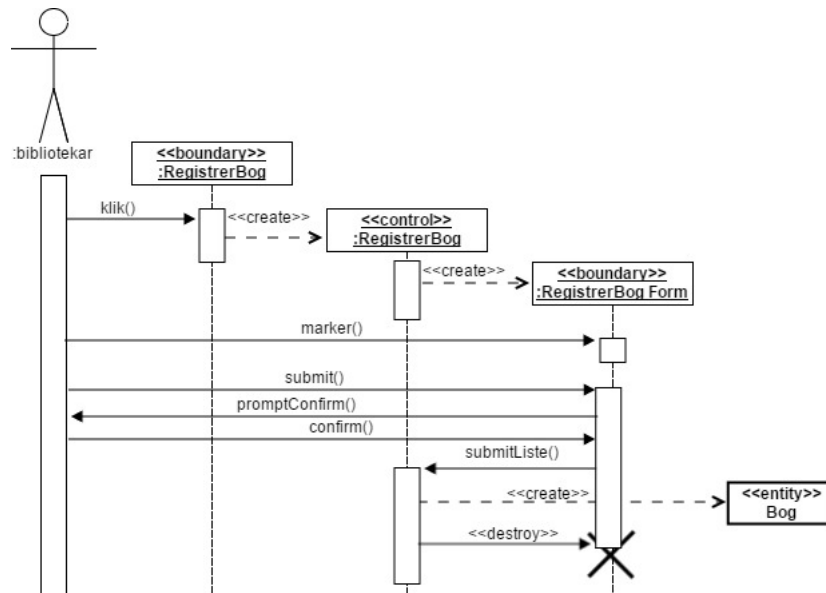


Fig. 3.7: Figuren herover et et sekvensdiagram over use casen der omhandler registrering af bog.

I denne figur 3.7, er sekvens processen ligeledes delt op i tre dele, menupunktet registrer bog, en controller, og formen der skal udfyldes. Herefter skal bibliotekaren indtaste data på bogen(hvad dette indeholder er ikke fastlagt, men det vil være ting som forfatter, år, titel osv.). Herefter bliver han bedt om at bekræfte at han har indtastet de rigtige informationer. Klikkes der her på nej, skal han starte forfra/redigere i det indtastede. Klikkes der bekræft, sendes inputtet til databasen, og den returnerer om hvor vidt bogen blev registreret.

3.6.3 Udsending af rykker

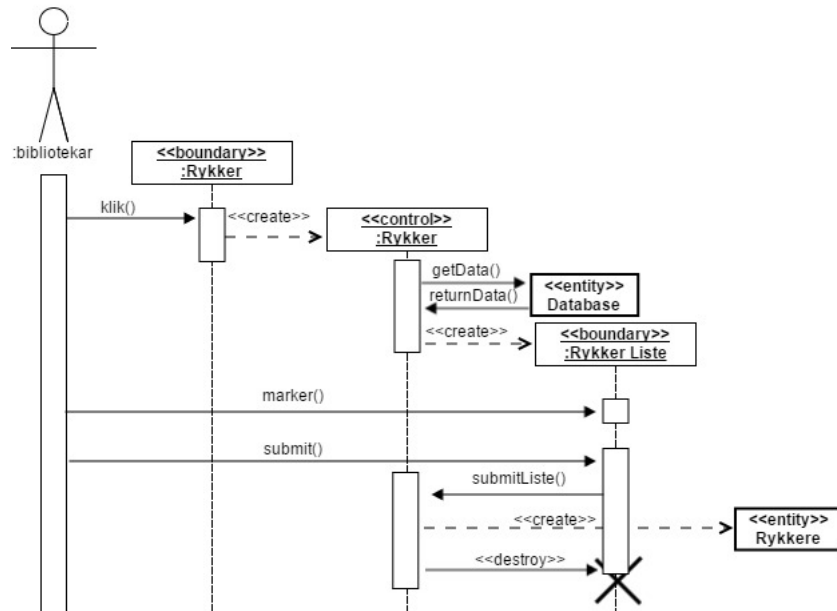


Fig. 3.8: Figuren herover et et sekvensdiagram over use casen der omhandler rykning af bog.

I figur 3.8 herover, er sekvens processen ligeledes delt op i tre dele, menupunktet rykker, en controller, og en liste der indeholder de udlån der har overskredet afleveringsdatoen. Denne data er hentet fra databasen. Her klikker bibliotekaren på knappen send rykkere, og dataen sendes til mail-serveren. Her sendes mails afsted, og den returnerer om hvor vidt de blev sendt eller ej. Til slut registreres de udsendte rykkere i databasen, dette er dog ikke med i diagrammet herover.

4 Systemdesign sammenfatning

4.1 Kort om systemet

Systemet skal fungere som en hjemmeside, der er programmeret i HTML, CSS og PHP, samt en mySQL database hvori alt data skal lagres. Hjemmesiden er kun tilgængelig fra bibliotekets eget netværk, således at man formindsker risikoen ved at nogle misbruger det. Dette gøres ved at kontrollere IP-adressen.

Systemet skal desuden være konstrueret sådan at bibliotekets ansatte har et fælles login, således at de har flere muligheder end de ordinære brugere af systemet, såsom at tilføje eller fjerne brugere og bøger fra systemet. Da dette login kræves, vil siden altså også kunne fungere som selvbetjening for den enkelte låner, da de kun skal have mulighed for at låne og aflevere bøger.

4.2 Deploymentdiagram

Et deployment diagram beskriver forholdet mellem komponenter og enheder(nodes). Komponenter er mindre enkeltstående systemer, som snakker sammen med andre komponenter. Så som en webserver der snakker sammen med en browser, hvor en browser snakker sammen med en bruger. En enhed skal forstås som et fysisk objekt så som en computer eller en webhost.[s. 256, kilde 1]

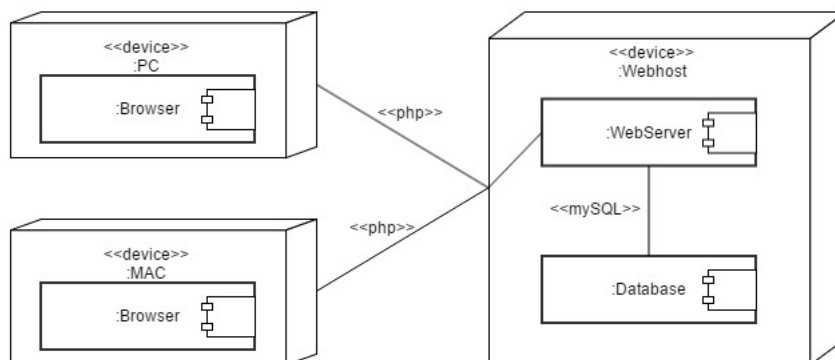


Fig. 4.1: Figuren herover er et deploymentdiagram over vores system.

Figur 4.1 herover illustrerer forholdet mellem en browser og den webhost, som siden vil ligge på. Der fremgår to tilsyneladende ens devices på diagrammet, PC og MAC, og dette er blot for at vise at systemet skal kunne køre på de to forskellige styresystemer. Vores webhost administrerer både webserver og database, hvor mySQL bliver brugt til kommunikation mellem disse. Forbindelsen og kommunikation mellem browser og webserveren på webhosten, foregår med serversidesproget php.

4.3 Datalagring

For at dette system skal kunne fungere, er der behov for en form for lagring af alt det data, systemet er bygget op omkring; bøger, brugere, rykkere og udlån. Denne data er vigtig for systemet og bør derfor gemmes forsvarligt,

og kunne tilgås uden de store komplikationer. Her forekommer der naturligvis nogle forskellige muligheder for lagring, heriblandt en simpel fil, hvori alt data ligger. Dette er dog ikke en optimal løsning, da det kan tage meget lang tid at finde data i en stor text fil.

Det vil altså give langt bedre mening at bruge en relationel database til data-lagring, da denne kan indekser sine data, således at man langt hurtigere kan finde det data man søger efter. Denne database vil ligge hos en webhost, som sørger for at der findes backups af databasen, hvis nu der var noget der skulle fejle.[s. 262, kilde 1]

4.4 Adgangskontrol

Da dette er et multi-bruger system og der findes to forskellige brugertyper, kræves det at man kontrollerer hvad den enkelte bruger skal kunne foretage sig i systemet[s. 263, kilde 1]. Ydermere skal det også sikres at systemet kun kan tilgås, fra bibliotekets netværk.

4.4.1 Adgang til systemet

Når man forsøger at tilgå systemet, vil det som det første kontrollere om brugerens nuværende ip-adresse, matcher bibliotekets. Hvis dette ikke er tilfældet, vil man blive nægtet adgang til bibliotekets system. Vi har brugt en indbygget funktionalitet i PHP, til at hive brugerens ip-adresse ud, for herefter at sammenligne den med bibliotekets.

4.4.2 Brugertyper og kontrol

Da bibliotekarere og lånere ikke skal have samme muligheder i systemet, kræves det en kontrol af hvem af de to brugertyper der tilgår systemet. Dette sikres ved at bibliotekarere har et brugernavn og adgangskode. Hvis dette indtastes og er korrekt, vil de kunne tilgå flere funktioner i systemet, hvorimod hvis der intet login er angivet, vil man blot have mulighed for at aflevere eller låne bøger. Dette er ikke implementeret i systemet endnu.

5 Program- og systemtest

Systemtest bruges til at kontrollere om systemet opfører sig, som det forventes at det skal. Det kan være fra enkelte funktioner, til datalagring der fungerer korrekt. Derudover kan man også lave test af brugervenligheden af systemet, ved at observere og interviewe brugere af det. Efter sådanne tests, vil vi vurdere resultaterne af dem, og ændre det der kræves hvis der er behov for det[s. 427, kilde 1].

5.1 Funktionelle tests

Vi har udført en række test på de funktioner der findes i systemet indtil nu, dvs. minimuskravene. Disse test er blevet udført løbende, i takt med udviklingen af de enkelte funktioner. Her har vi testet både, at systemet opfører sig overfor brugeren som det skal, altså vise de rigtige beskeder og give den korrekte feedback på forskellige handlinger. Hvis en bog eksempelvis allerede er udlånt, skal den ikke kunne udlånes igen før den er blevet returneret, og dette skal fortælles til brugeren der oplever fejlen. Derudover har vi også testet at de rigtige data lagres korrekt i databasen, ved brug af de forskellige funktioner.

Vi har også sørget for, og testet, at hvis en bruger har overskredet returneringsdatoen for en bog, skal vedkommende ikke spammes med et overflødigt antal rykkere. Det skal kunne begrænses til at de modtager en hver måned.

5.2 Test af brugervenlighed

Brugervenligheden af systemet har vi ikke fået testet endnu, men det vil blive testet ved at lave tænke højt forsøg med folk, der ikke i forvejen har kendskab til systemet.

6 Brugergrenseflade og interaktionsdesign

Herunder er vist skærbilleder af de mest interessante dele af vores brugergrænseflade og vores generelle design i vores første prototype. Vi lægger stor vægt på at systemet skal være så simpelt som muligt, og intuitivt hvorledes systemet fungerer. Der vil sandsynligvis senere blive skrevet nogle flere vejledende tekster til de forskellige funktioner, for at gøre det lettere for brugeren. Vi har også valgt at highlighte hvilken side på systemet, man er inde på, igen for at gøre systemet mere overskueligt og navigerbart. Som det ses på billederne er 'Lån bog' skrevet med tyk skrift når man er på den side, og 'Rykkere' når man er der.

6.1 Lån bog

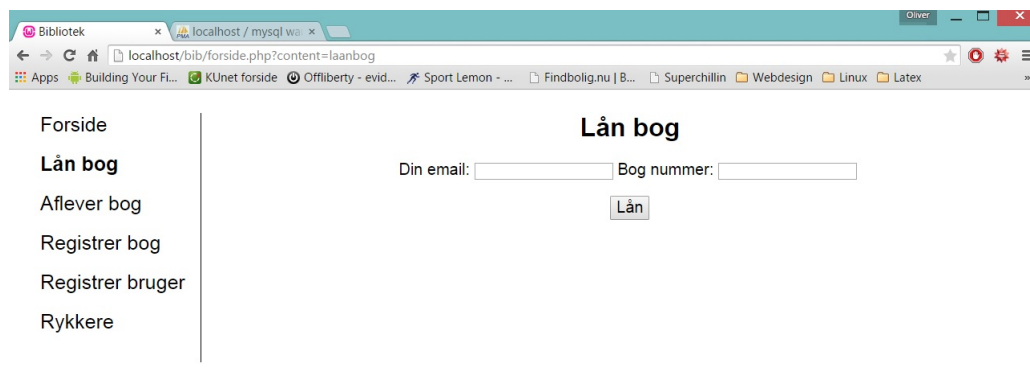


Fig. 6.1: Figuren herover er et skærbillede af siden 'Lån bog'.

Figur 6.1 viser hvordan vores underside 'Lån bog' ser ud. En bog lånes ved at indtaste de ønskede informationer i felterne og klikke på lån. Afhængig af inputtet vil følgende besked komme frem under knappen:

- Den indtastede mail findes ikke: 'Den indtastede mail er ikke registreret.'
- Den indtastede bog findes ikke: 'Bogen til det indtastede bognummer, findes ikke i systemet.'
- Bogen er allerede udlånt: 'Bogen er allerede udlånt.'
- Hvis inputtet er korrekt: 'Lånet er nu registreret.'

6.2 Rykkere

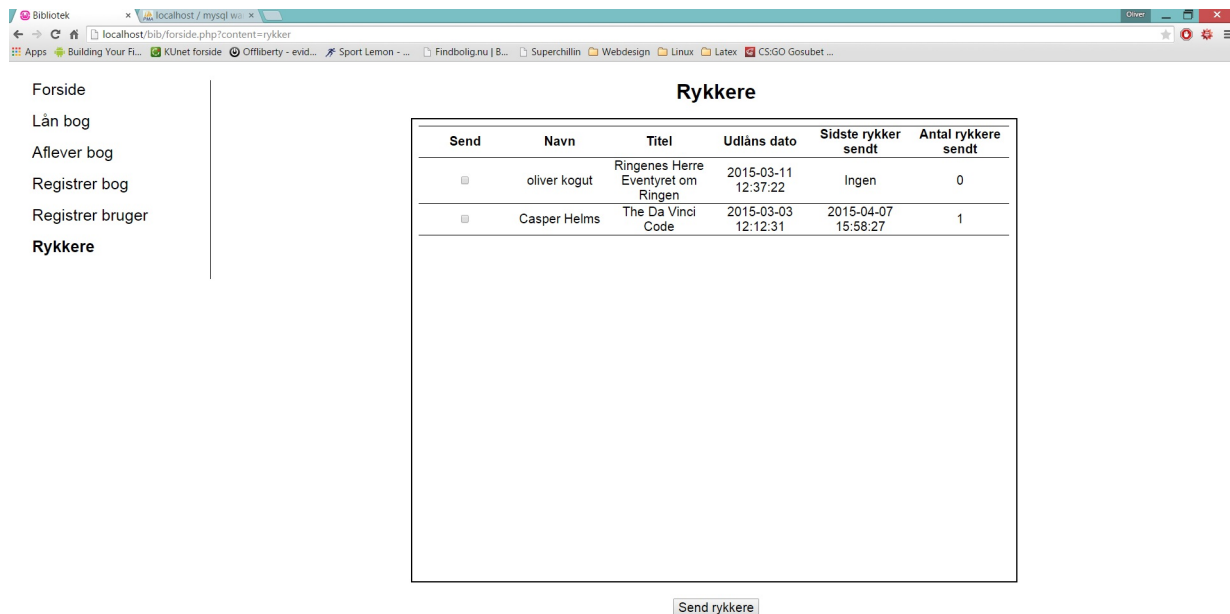


Fig. 6.2: Figuren herover er et skærbillede af siden 'Rykkere'.

Figur 6.2 viser hvordan vores underside 'Rykkere' ser ud. På denne side fremgår en liste over alle udlån, som har overskredet returneringsdatoen. Her kan en bibliotekar hakke de felter af i første kolonne, som vedkommende ønsker der skal sendes rykkere til. Et udlån der er overskredet vil kun fremgå af denne liste, hvis det er mere end 30 dage siden, der sidst blev sendt en rykker. Hvis der ikke har været sendt en rykker, vil udlånet fremgå hvis udlånet er overskredet med 30 dage. Der vises i øvrigt også antallet af rykkere der er blevet sendt. Hvis listen bliver så lang at der kræves mere plads end den box der vises, vil der komme en scrollbar frem, således at man kan rulle ned i selve boxen. Når bibliotekaren har markeret hvem der skal sendes rykkere til, klikkes der på knappen 'Send rykkere'.

6.3 Præsentation af kørende prototype

Vi har lavet en video præsentation af vores system, hvor alle dets nuværende funktioner bliver vist. Af en eller anden grund er linjen mellem udlån på rykker listen ikke kommet med. Videoen ligger som en skjult video på youtube, og kan ses på dette link:

Prototype 1: <http://youtu.be/53zYQDoB3ps>

7 Projektsamarbejdet

7.1 I gruppen

Arbejdet internt i gruppen fungerer ganske fint, selvom vi alle har været tidspresset udover dette projektarbejde. Derfor har vi valgt ikke at holde fysiske møder, og i stedet benyttet os af kommunikations- og skærmdelingsværktøjet Skype. Som udgangspunkt arbejder vi i gruppen sammen om tingene, men hvis der udføres individuelle opgaver, vendes resultatet med alle i gruppen, og diskuterer eventuelle rettelser.

7.1.1 Hvad går godt

Vores fremgang med udviklingen af systemet går godt, selvom vi kom lidt sent fra start. Vi er kommet godt efter det og har nu en fungerende prototype, som opfylder kundens minimumskrav.

Vi er gode til at samarbejde omkring udvikling af henholdsvis system og rapporter, og drøfter beslutninger og de forskellige muligheder der måtte opstå.

7.1.2 Hvad går skidt

Vi har haft nogle småproblemer rundt omkring, selvom det generalt går fint. Vi har blandt andet oplevet problemer med Github, da der til tider er problemer med henholdsvis at pushe og pulle ændringer, dette er dog noget vi formår at løse hver gang, så det er mest af alt bare irriterende og unødigt tidskrævende.

Derudover har vi opdaget at det kan være problematisk at rette en figur vi tidligere har lavet, da det er gemt som billede. Nogle af de figurer vi har, er lavet på www.glify.com og kan let redigeres, der er dog kun mulighed for at gemme 5 filer i den gratis version af programmet. Dette har også resulteret i irritation og tidsspild.

7.2 Kunden

Der er desværre opstået problemer i forhold til vores samarbejde med kunden, da de ikke længere svarer på hverken emails eller facebook beskeder. Derfor kan det ikke undgås at store dele af vores system vil blive bygget på antagelser, på hvad kunden måtte forlange at systemet skal kunne. Derudover har vores instruktør Markus, tilbudt at han kan agere kunde i situationer hvor vi har særligt behov for det, og eventuelt som tester af systemet.

8 Bibliografi

1. Bruegge, Bernd og Dutoit, Allen H. 2014. "Object-Oriented Software Engineering Using UML, Patterns, and Java third edition"
2. Mathiassen L., Munk-Madsen A., Nielsen P. A., Stage J. 2000. "Object-Oriented Analysis and Design" (Uddraget fra ugeseddel 6)
3. Gould, J. D. og Lewis, C. 1985. "Designing for usability: key principles and what designers think". s. 300-311
4. Parnas, D. L. og Clements, P. C. 1986. "A rational design process: How and why to fake it" s. 251-257
5. Sutherland, Jeff. 2010. "SCRUM Handbook"
6. Brooks, F. P. (1986), "No silver bullet", Proc. of the IFIP Tenth World Computing Conference
7. Michael Christensen, Andy Crabtree, Christian Heide Damm, Klaus Marius Hansen, Ole Lehrmann Madsen, Pernille Marquardsen, Preben Mogensen, Elmer Sandvad, Lennert Sloth, and Michael Thomsen. (1998). "The M.A.D. Experience: Multiperspective Application Development in evolutionary prototyping." In: Proceedings of the 12th European Conference on Object-Oriented Programming (ECCOP '98), Eric Jul (Ed.). Springer-Verlag, London, UK, 13-40.

9 Litteratur Review

Herunder er udført to litteraturreviews af to artikler. Artiklerne fremgår også af bibliografien.

9.1 The M.A.D. Experience

Vi har beklageligvis ikke fået lavet dette review. Det vil der naturligvis blive rettet op på til genaflevering.

9.2 No Silver Bullet - Essence and accident in software engineering

Denne artikel af Frederick P. Brooks - "No Silver Bullet - Essence and Accident in Software Engineering" omhandler problemerne ved software programmering og hvordan at der ikke kan findes et gennembrud der ville garantere en stor forbedring indenfor software udvikling. Han beskriver disse gennembrud som 'Orders-of-magnitude improvements' og refererer samtidig til hardware branchen som på det tidspunkt havde set "*6 Orders-of-magnitude price performance gain in the last 30 years*" [kilde 6, side 182].

Titlen refererer til en anekdote han forklarer omkring varulve og deres svaghed - en såkaldt 'silver bullet'. Han mener dog ikke at en føromtalt 'silver bullet' er at finde inden for software programmerings branchen. F.P. Brooks deler problemerne op i 2 typer: "essential and accidental" hvor at 'essential problems' er de forskellige data set, relationerne imellem data typer, algoritmer og implementering af funktioner, og 'accidental problems' er de problemer der opstår imens man skriver koden men som ikke har noget med selve softwaren at gøre. Derefter deler han de essentielle problemer op i 4 yderligere undergrupper: kompleksitet, konformitet, flexibilitet og usynlighed som han derefter analyserer.

Det samme gør han med de 'accidental problems' hvor de bliver delt op i 3 grupper: Gode programmeringssprog, 'Time-sharing' og rigtig valg af styresystem. Han nævner desuden også at alle store gennembrud inden for de accidental problems har været der, så vi skal ikke forvente noget stort lige foreløbigt.

I det næste afsnit kommer han ind på hvordan man i fremtiden vil kunne nærme sig en 'silver bullet' hvis der fokuseres på følgende emner inden for software udvikling: ADA og andre høj niveau programmeringssprog, objekt-orienteret programmering, AI (Artificial Intelligence), automatisk programmering og bedre værktøjer generelt.

Et emne som han vender tilbage til flere gange er brug af allerede implementerede moduler - "*The most radical possible solution for constructing software is not to construct it at all*" [kilde 6, side 197]. Han forklarer hvordan tiderne har ændret sig fra starten af 1950'erne da markedet for køb af software startede op og der ikke var særlig meget gang i markedet indtil nu (~1985) hvor de fleste er villige til at købe et forudlavet modul til deres software. I et senere emne beskriver han hvordan han selv i 1958 fik åbnet sine øjne op for at 'bygge' et program fremfor at 'skrive' det, men nu mener han at det er på tide at ændre det igen. Ligesom sidste rapports artikel "A rational design process: How and why to fake it" [Kilde 4] mener F.P. Brooks også at det er umuligt at planlægge et it-system i forvejen. Han skriver, at nuværende it-programmer skal laves ved 'grow, not build' konceptet som han beskriver på [kilde 6, side 201]. Ved inkrementering, bruger-tests og en iterativ arbejdsproces skal systemet opbygges først som en startende grov prototype og efterfølgende små dele ad gangen. Som en afslutning på hans artikel forklarer han hvordan rigtig gode programmører er sjældne og firmaer burde gøre deres bedste for at opnå deres fulde potentiale.

10 Bilag 1: Versionsstyring
















Link til github repository:





































<https://github.com/oliver3660/ProjDat2015.git>




























Vi har herunder indkluderet vores nuværende commit-log. Vi har dog ikke inkluderet commits helt fra starten af, da det ikke har haft indflydelse på projektet. Det var blot nogle tests til at lære github at kende.

Vores commit d. 20 April, var vores første commit af kode til systemet, og indeholdt blot det visuelle i systemet, altså ingen funktionalitet.

Vores commit d. 4 Maj, var vores første prototype til systemet, hvori alle funktionelle minimumskrav er opfyldt.

Commits on May 4, 2015	
 prototype + rapport 2 oliver3660 authored 7 days ago	 3f97d5e 
Commits on Apr 26, 2015	
 Ændringer til kodemappen v.3 kiksekage authored 15 days ago	 90bb355 
 ændringer til kodemappen kiksekage authored 15 days ago	 ce94764 
 Database fil kiksekage authored 15 days ago	 4d7775f 
Commits on Apr 21, 2015	
 sidste ændringer rapport 2 oliver3660 authored 20 days ago	 c068d56 

Commits on Apr 20, 2015
<div> kode1 oliver3660 authored 2 hours ago</div> <div> e93b9f8 </div>
Commits on Apr 19, 2015
<div> Rapport 2 - Opdatering oliver3660 authored 20 hours ago</div> <div> 80f95d1 </div>
Commits on Apr 15, 2015
<div> opdateret rapport 2 oliver3660 authored 5 days ago</div> <div> 24ac76d </div>
<div> tilføjer kodemappe oliver3660 authored 5 days ago</div> <div> 43ae3c5 </div>
<div> flytter readme oliver3660 authored 5 days ago</div> <div> e86fa9b </div>
<div> flytter Readme oliver3660 authored 5 days ago</div> <div> 8505682 </div>
<div> tilføjer mappe oliver3660 authored 5 days ago</div> <div> 5844c2f </div>
<div> Sletter alt for at rydde op oliver3660 authored 5 days ago</div> <div> 8df0455 </div>
<div> Slettet fil kopi oliver3660 authored 5 days ago</div> <div> be31901 </div>
Commits on Apr 14, 2015
<div> Rapport 2, start oliver3660 authored 6 days ago</div> <div> a7d090c </div>
Commits on Mar 30, 2015
<div> Merge branch 'master' of https://github.com/oliver3660/ProjDat2015 oliver3660 authored 21 days ago</div> <div> a8494eb </div>
<div> done rapport oliver3660 authored 21 days ago</div> <div> a433e09 </div>

Commits on Mar 26, 2015	<div> krav til delrapport 2 kiksekage authored 25 days ago</div> <div> 6f01597 </div>
	<div> jeg addede lige rettelser kiksekage authored 25 days ago</div> <div> eeda4b2 </div>
Commits on Mar 24, 2015	<div> review1 oliver3660 authored 27 days ago</div> <div> 983d0df </div>
Commits on Mar 22, 2015	<div> rapport oliver3660 authored 29 days ago</div> <div> db2743f </div>
Commits on Mar 19, 2015	<div> rapport oliver3660 authored on 19 Mar</div> <div> b603ae6 </div>
Commits on Mar 16, 2015	<div> rapport oliver3660 authored on 16 Mar</div> <div> 8b19022 </div>
Commits on Mar 13, 2015	<div> problemformulering oliver3660 authored on 13 Mar</div> <div> 60bb100 </div>
Commits on Mar 12, 2015	<div> nyt billede kiksekage authored on 12 Mar</div> <div> d850f85 </div>
Commits on Mar 10, 2015	<div> Systembeskrivelse oliver3660 authored on 10 Mar</div> <div> 9df1d3d </div>

11 Bilag 2: Changelog for projektrapporten

14-04-2015 OSK Rapport påbegyndt
14-04-2015 OSK Punkt 2 og 3 påbegyndt
15-04-2015 OSK Review af "A rational design process:..." færdiggjort
15-04-2015 Alle Punkt 7 påbegyndt
15-04-2015 Alle Punkt 1 påbegyndt
19-04-2015 Alle Punkt 3 fortsat
19-04-2015 TNS Review af "Designing for usability:..."
20-04-2015 Alle Punkt 1, 2, 3, 7 færdiggjort
04-05-2015 Alle Rapport rettet til genaflevering
11-05-2015 Alle Use-cases, BCE, sekvensdiagram beskrivelser rettet.
11-05-2015 Alle Punkt 4, 5, 6, 7, 10 rettet.
11-05-2015 Alle Punkt 9.2 lavet.

12 Bilag 3: Timeline

Tidslinjen herunder daterer og beskriver milepæle og/eller centrale igangsætninger og afslutninger af processer.

09-03-2015:

Møde med product owner Simon Shine, for at få de overordnede krav, og ønsker

til IT-systemet.

13-03-2015:

Arbejde med første delrapport begyndes.

23-03-2015:

Første delrapport færdig.

14-04-2015:

Arbejde med anden delrapport begyndes.

20-04-2015:

Kodning af systemet begyndes.

04-05-2015:

Prototype 1 færdig.