Modern nätadministration

– ett systemperspektiv

LUNDS
UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg

Examensarbete:
Joachim Andersson

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2010

Sammanfattning

Detta examensarbete handlar om utveckling av ett system för hantering av DNS och DHCP. Det handlar om hela flödet från hantering av krav från beställare, utvecklingsmetodik, databasmodulering och utveckling. Syftet med detta examensarbete är att utveckla en applikation som gör att man slipper personberoende som finns på den gamla applikationen. Möjlighet för vidareutveckling och utveckling av ett ramverk för framtida utveckling på LDC.

Rapporten går stegvist igenom tidsuppskattning, hur databas modellen modulerades, utbildning av Seam och ur utvecklingsarbetet genomfördes och med vilken metod.

Analyserar de problem som uppstått vid utvecklingen av applikationen, hur det blivit lösta.

Målet är med detta arbete är kunna användarna själva att administrera DNS och DHCP och även bygga grunden till vidareutveckling. Även tack vare utveckling av ett nytt ramverk som ska användas så kommer personberoende på applikationen försvinna.

Abstract

This thesis is about developing a system for managing DNS and DHCP. It's about the entire flow from the handling of requests from clients, development methodology, database modulation and software development. The purpose of this paper is to develop an application that will eliminate the person dependency on the application. Possibility for further software development and development of a framework for LDC.

The report goes step by step through the time estimate, how the database model is modulated and education of Seam and how development work was carried out and by what method.

Analyzing the problems encountered in the development of the application and how it was solved.

The goal is of this thesis is to make the users manage DNS and DHCP by themselves, and also build the foundation for further development. Thanks to the development of a new framework, the person dependency is gone.

Förord

Detta arbete är ett examensarbete inom ramen för programvaruteknik., LTH Ingenjörshögskolan vid Campus Helsingborg. Omfattningen är av examensarbetet är 22.5 högskolepoäng. Arbetet utfördes på uppdrag av Lunds Universitet.

Arbetet initierades då LDC behövde får en applikation som funnits hos en extern part och man ville ha applikationen i huset för att lättare kunna göra ny utveckling. Det bestämdes då att den skulle utvecklas i Java och SEAM.

Jag är väldigt tacksam för möjligheten till detta och har lärt mig väldigt mycket inom Java, Seam, Hibernate och SQL.

Jag skulle vilja tacka min handledare Marika som har hjälpt och haft tålamod med mig undertiden, vill även tacka mina arbetskamrater och framförallt min Team Ledare Niklas Löfgren som gjort det möjligt för mig att slutföra min utbildning.

Innehållsförteckning

Inledning

Detta examensarbete är en avgränsad del av ett större projekt som drivs på LDC. LDC är en del av den centrala förvaltningen på Lunds Universitet och har ansvar för de centrala systemen på Universitet. T.ex. Mail, Nät till datorerna och Ekonomisystem. LDC bytte namn från Lunds Data Central 2006. I detta arbete kommer en webbaserad applikation byggas för hantering av DNS och DHCP, användargränssnittet kommer att begränsas till den delen som administrativ personal på LDC kommer använda sig av. Applikationen ska även kunna generera de filer som används av DNS och DHCP servrarna som finns på LDC. Där ska finnas felhantering så att det inte kommer föras över några felaktiga filer och även en nattlig kontroll på diversesaker t.ex. att alla cnamn är korrekta, namnet som är från ldap är uppdaterade. Applikationen ska även ha stöd för flera språk som kommer vara en del av ramverket.

Denna applikation har utvecklats då LDC vill ha kontroll över koden och möjlighet att i framtiden kunna lägga till funktionalitet. Projektet CFL har krav på att man i framtiden ska kunna hantera IPV6, switchar och telefoni via NetTools.

I detta projekt ingår även utvecklingen av ett ramverk som ska användas vid utveckling av andra applikationer på LDC. Detta görs då de ska ha möjlighet att snabbt komma igång vid utvecklingen av nya applikationer.

Mina två handledare är Marika Cochinescu och Paul Hedberg. Marika är projektledare för projektet och Paul är systemutvecklare på LDC.

1 Bakgrund

De senaste åren har LDC haft i uppdrag att uppgradera stamnätet för hela Lunds Universitet. Detta projekt har gått under namnet CFL(Centralt Funktionsansvar för Lokalt nät). Detta projekt innebär ett centralt ägande och drift av de lokala fastighetsnäten inklusive all nätverksutrustning för data- och telekommunikation.

I detta CFL igår att Lokal IT-personal ska kunna ha möjligheten att hantera deras del av DNS, DHCP, telefoni- och switch- inställningar. Idag finns ett system som tar hand om hantering av DNS och DHCP som man vill utöka till att kunna ta hand om telefoni- och switchinställningar och även utöka med ytterligare funktion för DNS och DHCP.

Problemet som finns idag är att LDC har ett system som utvecklas och ägs av Banhof. När det behövs ändringar och anpassningar efter Lunds Universitets behov så tar det tid och går inte alltid att anpassa på grund av att det finns bara en utvecklare på Banhof som har hand om detta system. LDC vill bli av med personberoendet på system och vill därför utveckla ett eget system där de kan utbilda sina egna utvecklare.

Detta har lett till en beställning av ett nytt system som ska utvecklas hos LDC för ökad kontroll av systemet. Beställaren av detta är Gunnar Knutsson på LDC. Gunnar är tjänsteansvarig för CFL.

1.1 Beskrivning av nuvarande system

Dagen system, LUDD, möjliggör för lokala dator- och nätverksansvariga att själva uppdatera information om IP-nummer för nya eller utbytta datorer och nätverkskort, vilket förenklar hanteringen av den centrala IP-nummerhanteringen.

Systemet anskaffades 2005 från Bahnhof och LDC äger fulla rättigheter till de delar som levererats och som används för eget bruk. Efterhand som nya önskemål har kommit in från användarna har systemet förbättrats med hjälp av konsultinsatser från Bahnhof. Kunskapen om systemet är begränsat till en fysisk person på Bahnhof och detta har på senare tid medfört långa leveranstider när den här personen har varit fullbelagd med andra uppdrag. Svårigheten att få tillgång till konsulttid för utveckling och support upplevs som en stor risk.

1.2 Styrgrupp

Alla beslut som kommer att påverka kostnaden av utvecklingen av NetTools kommer att tas av styrgruppen, de kommer också ha som uppgift att godkänna slutprodukten.

Styrgruppen för detta utvecklingsarbete är:

- Gabriela Kalus, Datachef för Fysikum.
- John Westerlund, IT-strateg
- Jan-Inge Månsson, Avdelningschef för kund och tjänst på LDC
- Gunnar Knutsson Tjänsteansvarig på LDC

1.3 Projekt

Detta examensarbete är en det av ett projekt som drivs på LDC. I detta projekt ingår följande personer.

- Marika Cochinescu, Projektledare.
- Paul Hedberg, Systemutvecklare
- Joachim Andersson, Systemutvecklare
- Lars Holst, tekniskrådgivare för DNS
- Paul Haglund, tekniskrådgivare för DHCP

Marikas uppgift som projektledare är att följa upp så att projektet följer den tidsplan som är uppsatt, rapportera till styrgruppen om problem som uppstår. Marika är mentorer för det här examensarbetet.

Paul Hedberg är systemutvecklare på LDC och kommer att utveckla de delar för NetTools som inte ingår i detta examensarbete.

Lars Holst och Paul Haglund jobbar på LDCs avdelning Kommunikation och kommer fungera som teknisk hjälp och rådgivning när det gäller frågor angående DNC och DHCP.

1.4 Mål

I detta examensarbete kommer att vara med att ta fram ett ramverk som kommer användas vid utveckling av NetTools och för framtida användning av utvecklingsprojekt i Java på LDC. Ramverket kommer att baseras på Seam 2.2.1[1] som är utvecklat av Redhat och är ett opensource projekt.

[1] http://seamframework.org/

Det ska utvärderas om den existerande databasen stöder examensarbets krav eller om det ska designa en ny databasmodell. Utveckla ett gränssnitt för den Administrativa Personalen på LDC. Övrig gränssnittsutveckling är inte en del av detta examensarbete. Examensarbetet kommer till största delen att fokusera på utvecklingen av applikationen.

Det slutgiltiga målet är att ha en applikation där man kan hantera DNS och DHCP via ett gränssnitt som är anpassat för den administrativa personalen på Lunds Universitet. Det ska även vara möjligt att ta in ytterligare kunder som ligger utanför universitet.

- Databasmodulering. Går det att använda den existerande databasmodellen eller ska den moduleras om.
- Överföring av data från gamla databasen till den nya databasen.
- Kontroll så data är rätt överförd.
- Gränssnitt för administrativ personal på LDC.
- Generering av DHCP och DNS filer.
- Utveckling av ramverket.
- Fellogg, så som vid felaktiga parametrar i DHCP, DNS eller felaktiga cnamn etc.
- Dagliga kontroller av att t.ex. cnamn är giltiga.

1.5 Problemställning

Det viktiga med de punkterna i problemställningen är att kunna få ett lätt sätt att vidareutveckla applikationen, lätt för övriga utvecklare att förstå hur uppbyggnaden är och få en stabil modell.

- Utvärdera om den gamla databasmodellen är möjlig att använda för vidareutveckling eller om man måste skapa en ny modell som stöder beställningens krav bättre.
- Sätta upp ett ramverk som till större delen kan återanvändas av applikationer som senare kommer utvecklas av LDC. Det ska även vara lätt att särskilja på ramverkets delar så som de delar som är specifika för just den här applikationen.
- Utveckla ett gränssnitt för den administrativa personalen på LDC som uppfyller de krav som ställs på den.
- Generering av DHCP och DNS filer så de passar det formatet som används av Lunds Universitets DNS och DHCP Servers.

1.6 Avgränsningar

Dessa avgränsningar kommer inte genomföras i detta examensarbete, det ska dock vara möjligt att utföra dessa avgränsningar efter examensarbetet är utfört.

- Ramverket kommer bara utvecklas i det här examensarbetet till den delen så att det passar till detta examensarbete. Där kommer finnas funktioner i ramverket som kommer behövas i framtida projekt på LDC som inte kommer utvecklas i detta examensarbete.
- Gränssnitt för Lokala IT-ansvariga. Det gäller de som loggar in med enbart rollen User. Detta kommer att utvecklas av annan personal på LDC.
- Uppdatering och datahämtning från switchar eller andra utomliggande system.
- Telefonin. Det är planerat att hantering av IP-telefoni ska ligga i NetTools, dock har detta inte med detta projekt att göra då det kommer vara en av de delar som ingår i vidare utvecklingen efter detta examensarbete.
- Val av teknik för utveckling och plattform. Detta är något som har genomarbetats på LDC ett år innan utvecklingen av NetTools på började. D.v.s. val av databasserver, driftmiljö utvecklingsverktyg och även att Seam kommer att användas som grunden för utvecklingen.
- Överföring av filer till DNS och DHCP server. NetTools kommer att generera alla filer som behövs. Att överföra dem från servern där NetTools körs och läsa in dem i DNS respektive DHCP servern kommer att skötas av Team Kommunikation på LDC.
- Testning av systemet som blir utvecklat kommer att utföras av beställaren och de två tekniska rådgivarna, som dessutom är en del av slutanvändarna som kommer att använda NetTools.

1.7 Samband med andra projekt

1.7.1 CFL

Fastighetsnäten i Lunds universitets hyrda lokaler har tidigare ägts och administrerats av verksamheterna själva. Detta har medfört att kvalitén på fastighetsnäten har en mycket varierad standard. Projektet för övertagande av lokala fastighetsnät, CFL, innebär ett centralt ägande och driftansvar av dessa nät så att en enhetlig standard uppnås. Vid övergång till CFL har verksamheter framfört önskemål om att själva kunna utföra lokal patchning och konfigurering av switchar.

1.8 Tidsuppskattning

Innan projektet började så delades de olika delarna av implementationen upp i olika delar och tidsuppskattade dem enligt följande. Tidsuppskattningen [Figur 1] gjordes efter punkt 2.3 där det bestäms om det behöver genereras en ny databasmodell eller man kan användas sig av den exciterande.

	Timmar
Applikationsmiljö: Uppsättning/Konfigurering (Jboss AS, MySQL, ev. demo)	8
Implementering av KärnFNC(Ramverket) i Java (inlogg m.m.)	120
Inrätta entiteter i Java, tabeller i MySQL	30
Sätta upp gränssnitt för DB-uppdatering	10
DB-populering	40
Serverbeställning *2	0,5
Skapa DNS / PTR och DHCP fil	50
Skapa en/flera ”huvudkunder” för domän, nät. Arv till kunderna under.	20
Skapa olika grupper eller kunder, för varje institution	6
Tilldela kund en under-domän	15
Tilldela kund nät med möjlighet att ange VLAN, DHCP funktioner för nätet	6
Tilldela kund en användare	70
Bulk-ändra allt som kan ändras i vanliga fall	15
Flytta objekt mellan olika institutioner/kunder	10
Migrera alla objekt i ett nät till ett nytt nät	5
Ändra behörighet på objekt så att ”vanlig användare” inte kan ändra på ett objekt	8
Skapa DHCP grupper med olika DHCP optioner. Kunna tilldela sådan grupp till kund	15
Skapa en pool med IP-adresser som delas utav DHCP	20
Tilldela varje unikt objekt enskilda DNS/DHCP funktioner	10
Få fram en lista på alla nät som används.	10
Få fram en lista på alla vlan som används.	15
Ge info till olika institutioner eller alla institutioners användare när de loggar på sig	10
ge info till olika institutioner eller alla institutioners användare när de loggar på sig	5
Se DNS och DHCP filen som den kommer att skapas (felsöka)	15
Cname Editering	15
Aname Editering	15
Se log om senaste ändringarna och en varning om något i systemet ej fungerar	40
Utföra test på CNAME (nslookup)	5
Skapa headerfiler	10

Figur 1: Tidsuppskattning för utveckling av applikationen, tabellen innehåller antal timmar som är uppskattat för utveckling av de olika modulerna.

Totalt uppskattades utvecklingstiden till 598,5 timmar. Uppskattningen var lite osäker då vi inte utvecklat något i ramverket Seam innan. Den slutliga utvecklingstiden blev 460,25 timmar. Att det blev så pass stor skillnad

gällande utvecklingstiden för den delen Examensarbetet omfattar är på grund av att Seam ramverket gick snabbare att utveckla i än väntat. Det hjälpte till en stor del efter utbildningen av Seam att få fart på utvecklingen, hade kursen hållts tidigare i projektet kanske utvecklingstiden blivit ytterligare mindre. Denna tid omfattar dock inte den tid som är nerlagd på utbildning och litteraturläsning.

1.9 Förberedelser

Innan utveckling av produkten påbörjades var man tvungen till att lära sig hur Seam fungerar. De förberedelser som gjordes var att man läste 3 olika böcker[12][13][14] som gick igenom hur man utvecklar i Seam. I dessa böcker gick de igenom de grundläggande sakerna hur olika komponenter fungerar. En bit in under utvecklingen hölls även en 3-dagars kurs i Seam som anordnades av RedPill ett konsultföretag som har inriktat sig på RedHats produkter. Utbildning gjorde att man fick en större förståelse för Seam, detta lede till att utvecklingstiden blev lägre då man både blev effektivare och mindre felrättning.

1.10 Godkännande av slutprodukten

Produkten kommer att godkännas av styrgruppen och kunden. Den måste innan de godkänner den gå igenom tester av de två tekniska rådgivarna till projektet som kommer utvärdera den data som NetTools genererar till DNS- och DHCP-servrarna.

Vid drift av produkten kommer det att testas så att nätet på Lunds Universitet fungerar. Även att all information från förra systemet överensstämmer med det nya systemet. Om detta inte kommer att fungera så kommer inte produkten att godkännas utan man kommer att återgå till det gamla systemet tills alla problem i NetTools är lösta.

2 Metod

2.1 Utvecklingsmetodik

Utveckling av NetTools har delats upp i ett 3 delar. Delar före utveckling, utvecklingsfasen samt test och driftsättning.

Under den första delen som är före utvecklingen så gås kravspecifikationen igenom och diskuteras för att komma överens så att alla inblandade enas om kraven. Kraven är skrivna av beställaren och projektledaren och formuleras sedan om så att både kund och projektmedlemmarna förstår dem.

När kraven är färdigställda så tidsestimerar utvecklarna tillsammans en uppskattad tid om vad var och en av de olika funktionerna kommer ta att utveckla. I denna rapport kommer endast estimering av den tiden som tillhör examensarbetet att specificeras. När denna del är färdig och tidsuppskattningen är godkänd av kunden, alternativt att kunden stryker något krav för att budgeten ska hållas.

Den andra delen innefattar all analys och utvecklingsarbete.
Det första som gjordes var att utvärdera om man skulle använda sig av den existerande databasmodellen eller om det skulle designas en ny modell.

I detta projekt används testdriven utveckling, under utvecklingen så utgås det ifrån en aktivitetslista för både kontroll och uppföljning. Aktivitetslistan är alla de krav som är ställda av kunden, och under utvecklingen så delas aktiviteterna upp för att lättare kunna följa hur det går. Under utvecklingen är det daglig uppföljning med projektledaren om hur det går och ifall det är några problem.

Den tredje delen av utvecklingen står av sluttest, piloter och driftsättning.

När utvecklingen är klar så körs ett sluttest av projektets testare som även är de tekniska rådgivare för DNS och DHCP. Under denna fas så är det interaktiv process, mellan test och buggfixar tills man uppnår förväntat resultat.

Efter detta är det en liknande process som innefattar ett par piloter som kör systemet, dock i testmiljö då det inte är möjligt att köra 2 olika system för generering av filer (ersätter ett gammalt system som ska gör samma sak).

När alla tester är slutförda så är det upp till beställaren att godkänna produkten och bestämma ett datum för driftsättning. Vid driftsättning så planeras det både vad som ska göras vid driftsättning och vad som ska göras om driftsättningen fallerar.

2.2 Databas

Eftersom det ska göras om är ett gammalt system finns det även en databas för det gamla systemet. Det övervägdes om det skulle använda den gamla databasmodellen kontra att designa en ny databasmodell. Målet med projektet är bland annat att det ska bli enklare för framtida utveckling och kompetensöverföring av systemet så det inte står och faller med en person.

Fördelarna med att använda existerande databas är att det inte behövs någon överföring av data till den nya databasmodellen. Det sparas tid på att inte modellera en ny modell, den är relativt enkel att lägga till nya fält. Den är väldigt generell och LDC kan använda denna modell för de flesta program LDC ska utveckla i framtiden.

Nackdelarna med den är att den är relativt svår att sätta sig in i då den är uppbyggd på 2 tabeller, en tabell för själva objektet och en tabell för alla attributen som ska tillhöra det objektet. Den var inte den bästa modellen för att använda för Hibernate som LDC använder som API för databashantering, då den inte var uppdelat per objekt. Man behöver lägga mer logik i applikationen för hantering av hur objekten ska populeras.

Det beslut som togs var att i NetTools ska det skapas en ny databasmodell som både är lättare att förstå, mer anpassad till själva applikationen och mer relaterad till det som ska utvecklas. Även så underlättar detta förståelsen av uppbyggnaden av applikationen då man kan läsa ut de olika objekten som finns direkt ur databasen. Det som gjordes först var att specificera alla objekt som behövds, detta gjordes med hjälp av både Lars, Paul som hade den tekniska kunskapen angående DNS och DHCP om vilka typer som förekommer där.

De objekten som kan härledas från DNS är följande:

- DomainName
- Dnsgroup
- Cname
- HostName

I DHCPen hittades följande:

- Net
- Ipaddress
- Ippool
- MacAddress
- VLAN
- Dhcpgroup
- Dhcpconfiguration

Det behövs även en rad stödtabeller till dessa, först och främst till användarna och den tabellen kallas LuddUser och till den kommer det även att finnas roller. För att kunna dela upp grupper till vem som har rättigheter till att göra vad finns där ett objekt som heter Customer.
Fel behöver loggas när de inträffar och den tabellen heter Error.

2.2.1 LuddUser
Detta är själva användartabellen, i denna finns det information till vilka roller en användare har. Det finns olika roller beroende på vad användare får göra. Användare är kopplad till en eller flera Customers och på detta viset definieras vilka nät och domäner användaren har rättigheter till att ändra. En användare som är superadministratör kommer ha rättighet till alla Customers. Här sparas även när de var senast inloggad, ens fulla namn och även ett lösenord som bara kan användas för de personer som inte loggar in via Lunds Universitets centrala inloggningstjänst CAS.

2.2.2 Customer
Customer är själva kundobjektet och som det är på universitetet är det bundet till en institution, LDC har ett fåtal externa kunder. Kundobjektet innehåller ett antal domäner och nät som den kunden är ansvarig för och ska då kunna ändra i dessa. Där är även kopplat ett antal användare hit som i applikationen ska kunna hantera denna kund. Superadministratörerna vill kunna dela ut olika sorters DHCP inställningar som ska kunna tilldelas till specifika IP-adresser och där är också kopplat vilka DHCP-grupper en viss kund har tillgång till.

2.2.3 Error
Detta är en simpel lista med fel. De kan vara kopplade till ett visst objekt i databasen, detta är dock ingen hård koppling utan bara referens id till objektet.

2.2.4 DomainName
DomainName är det övergripande i DNS och är själva domännamnet, t.ex. ldc.lu.se. Detta objekt innehåller Cnamn och Hostnamn. Domännamnet är även kopplat till en Dnsgroup och en kund. Alla domännamn måste vara kopplat till en Customer.

2.2.5 Dnsgroup
Dnsgroup är en grupp av inställningar som kan tilldelas till ett domännamn. Den innehåller saker som TTL, SOA inställningar etc.

2.2.6 Cname
Cnamn är ett alias och detta är direkt kopplat till ett domännamn. Det innehåller namnet på alias och var det ska peka, ett exempel på detta är t.ex. www.ldc.lu.se som pekar på www.lu.se.

2.2.7 HostName
Ett hostname innehåller information om lite olika records i DNSen. Det innehåller information för A, SRV, MX records. Detta är kopplat direkt till ett domännamn, samma som med cname. Ett hostnamn kan vara kopplat till en eller flera ipadresser och för var och en av dessa kopplingar så motsvarar det ett A record. T.ex. att EGAWS3330 är kopplat mot 130.235.40.187.

2.2.8 Net
Ett nät är kopplat till en kund. Nätet är representation på ett nät i DHCPen, t.ex. 130.235.40.0/24. Nätet innehåller ett antal IP-adresser, en dhcpgroup för DHCP inställningarna. Till ett nät är även de macadresser som finns på nätet kopplade, detta för inte riskera att där blir dubbletter av någon macadress på ett nät.

2.2.9 Ipaddress
Ipaddress är representation av alla IP-adresser som finns i ett nät. Det är bara de använda adresserna som finns med i denna tabell, finns inte den med räknas den som oanvänd. Var användares IP-adress måste finnas med i listan då man inte använder dynamisk utdelning av IP-adresser för det fasta nätet, det behöver även i dessa fallen vara knutet en eller flera Mac adresser till en IP-adress. Tabellen innehåller även saker som vem som är kopplat till denna adress, vilka hostnamn som är kopplade hit.

2.2.10 Ippool
Ippoolerna är avsedda för IP-adresser som inte är avsedda för en specifik Mac adress utan med för t.ex. trådlösa nät där vem som helst ska kunna ansluta. En IP-pool innehåller ett visst antal IP-adresser som är knutna till just denna pool.

2.2.11 MacAddress
En Macadress är kopplat till både en IP-adress och ett nät. Det krävs 2 kopplingar här för att inte riskera att man har dubbletter av en Macadress på ett nät. Kopplingen mot en IP-adress är för att ange vilken Macadress som ska få motsvarande IP-adress.

2.2.12 VLAN
VLAN används för att ge användarna tillgång till sitt specifika nät. Varje nät är kopplat till ett VLAN. Detta är bara en tabell som används för information. Den har ingen praktiskt effekt i DNS eller DHCPen.

2.2.13 Dhcpgroup
Dhcpgroup är en samling inställningar man kan tilldela till IP-adresser och nät. I dhcpgroup finns ett visst antal fasta parametrar. Man kan även lägga till en eller flera dhcpkonfigurationer som är fritext för DHCP-inställningar.

2.2.14 Dhcpkonfiguration
Används som fritext för DHCP-inställningar och måste alltid tillhöra en dhcpgroup.

2.3 Ramverk

2.3.1 Teknisk specifikation
Innan detta projekt bestämdes vilka komponenter som skulle användas för utveckling av detta system samt LDCs ramverk, samt några tilläggspaket kom till under utvecklingen.

De som var ett krav för att utveckla i Seam är följande:

- Java 6 SE
- Seam 2.2.1
- Hibernate 3.3.1
- JSF 1.2
- Richfaces

Vid implementation så kommer även dessa 2 paket användas som är en del av Hibernate familjen, det nämns senare i detta kapitel varför dessa används.

- Hibernate Nverse
- Hibernate Search

Applications server som används är:

- Jboss 5.0.1 GA

2.4 Implementation

2.4.1 Ramverk
Själva utvecklingen av programmet kommer ske i ett ramverk som heter Seam framework[2] och är utvecklat av RedHat[3]. Den versionen som kommer att användas är version 2.2.1.CR1. Detta är ett ramverk som knyter samman olika färdiga API. Några av de komponenterna som kommer att användas är

[2] http://docs.jboss.org/seam/2.2.1.CR1/reference/en-US/html_single/
[3] http://www.redhat.com/

Hibernate[4], Jsf[5], RichFaces[6], Facelets [7]och Quartz[8]. Programspråket som kommer att användas är Java 6.

För att systemet ska fungera så behövs det utveckling av vissa stödfunktioner. De funktionerna som är ett krav för systemet är att det ska klara av hantering av flera språk, versionshantering av ändring i databasen, rollhantering och inloggning, stödfunktioner mot Lunds Universitets Ldap, hantering av menyer i systemet och sökning mot databasobjekten.

2.4.1.1 Seam

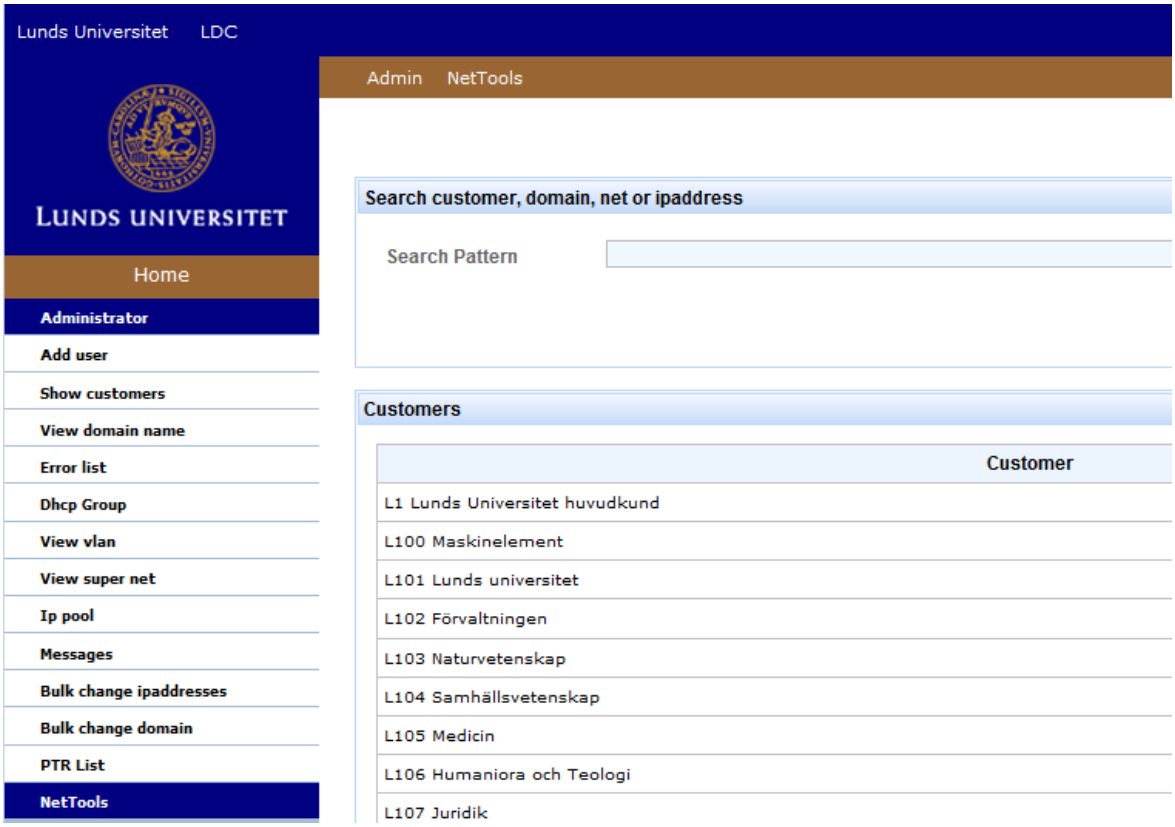
Orsaken till version 2.2.1 av Seam är bland annat dessa: I början på projektet så hölls en kurs i Seam. Utbildningen var i version 2.2.1.

Det är även för framtiden då Seam håller på att utveckla en ny version av Seam, Seam 3.0. Denna kommer att vara bakåtkompatibel med den senaste versionen av Seam 2. En annan anledning till att använda Seam 2.2.1 är att det kommer en version av 2.2.1 som ska fungera på Jboss Application Server 6.0. och då få möjlighet att använda nyare version av de flesta komponenter som ingår i Seam. Framförallt Hibernate 3.5 eller senare som kräver att man använder Jboss AS 6.0. Med senare version av Hibernate så kommer LDCs ramverk att få tillgång till fler funktioner när det gäller sökning och versionshantering.

2.4.1.2 Menyhantering

Menysystemet är uppbyggt i 3 nivåer för rendering av menyerna. Det är uppbyggt i en trädstruktur där den översta nivån som kallas applikations- nivå, som ligger överst i layouten, är kopplat till 0 eller flera menyer som man kan se tillvänster i LDCs applikation. Var och en av dessa menyer har ett menyobjekt som är själva länken till nya sidor. Var och en av dessa nivåer är kopplat till en roll. D.v.s. har inte användaren rollen kommer han inte se de respektive objekten i layouten. Det har dock inget med att göra om den har access till själva sidan utan det ställs in på sidnivå och inte i menyn, då man kan skriva in hela länken till sidan direkt utan att använda sig av menyn. [Figur 2]

[4] http://www.hibernate.org/
[5] http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html
[6] http://www.jboss.org/richfaces
[7] https://facelets.dev.java.net/
[8] http://www.quartz-scheduler.org/



Figur 2: Visning av menyn

2.4.1.3 Språkstöd

I ramverket är där ett inbyggt språkstöd för flera språk. Dock uppfyller inte detta riktigt de kraven det fanns på applikationen och för framtida bruk. Problemet med det inbyggda språkstödet är att det använder sig av resourcebundels och på grund av detta så kan man inte ändra fälten när applikationen är igång och det blir svårare att göra ett gränssnitt. Det som gjordes var att lyfta ut själva resurcebundel delen och bytte ut den så den läser mot en databas istället och utnyttjas här även den inbyggda cashningen som finns i Hibernate och Seam för databasanrop, så det kommer inte göras en databasfråga varje gång man ska få fram text för sidorna. Samtidigt som detta gjordes kopplades varje fält till en roll. Detta gjordes då det ska vara möjligt för kunden av applikationen att själv kunna ändra vissa, om de tycker att texten inte stämmer.

2.4.1.4 Inloggning och roller

På Lunds Universitet finns det en central inloggningstjänst som kallas CAS(Central authentication service) som ska användas vid inloggning. Detta är en produkt som är utvecklad av Jasig[9]. Fördelen med att använda denna är

[9] http://www.jasig.org/

att man bara behöver logga in en gång och sen kan man byta mellan system som använder sig av denna tjänst. CAS är kopplat mot Lunds Universitets Ldap och hämtar både användarnamn och lösenord från denna. Detta innebär att alla som är studenter, doktorander eller anställda kan logga in via denna tjänst.

Från CASen kan det hämtas ut vilken användare som är inloggad i systemet och detta knyts ihop med objektet LuddUser. Via objektet LuddUser kan man hämta ut vilka roller en användare är kopplad till.

Där finns även en annan typ av inloggning. Den är för användare som inte tillhör universitet och inte har något konto i Lunds Universitets Ldap. Denna typ av inloggning går via lösenordet som finns på LuddUser objektet och bara om det finns något lösenord på detta objekt. Denna typ av inloggning krävs pga att LDC även administrerar ett fåtal domäner och nät som inte tillhör Lunds Universitet.

Där finns 4 roller i NetTools:

- Administrator
- User
- Tekniskplattform
- LogViewer

Administratör är rollen som används av de som jobbar på LDC-avdelning Kommunikation och har fulla rättigheter i systemet.

User är vanliga användare som är lokala administratörer ute på de olika institutionerna och de ska bara kunna hantera en begränsad del av de nät och domäner som är tilldelade till just dem.

Teknisk plattform är till för LDC-avdelning Teknisk plattform som har lite mer rättigheter än en vanlig användare. De vill t.ex. kunna se en lista av alla nät som finns och vilka VLAN de tillhör.

LogViewer är en extra roll för Administratörer som är till för att visa loggarna på de olika objekten.

2.4.1.5 Sökning mot Lucat/Ldap

Lucat är den implementation och det gränssnittet Lunds Universitet använder sig för hantering av Ldap. Ldapen är en central roll i detta system då det är här man kan hämta ut information om de olika personerna som har loggat in i

systemet. Det utvecklades funktioner för att hämta information om de olika användarna, deras namn, epost etc.

Då man ska ha möjlighet att sätta attribut i Ldapen via NetTools utvecklades även funktioner för detta. Det är möjligt via NetTools att sätta att en viss användare ska hamna på sitt ”hemmanät” när de loggar in på det trådlösa nätet. Istället för att hamna på det generella trådlösa nätet som man annars hamnar på. Dessa funktioner är gjorda så de ska kunna återanvändas på vilken ldap man än vill ansluta sig mot och även för andra system som behöver den funktionaliteten.

2.4.1.6 Versionshantering

Vid hantering av versioner i systemet användes ett system som är en delkomponent till Hibernate. Detta heter Hibernate Envers och det som gjorts är att markera alla de objekten som ska version hanteras med annoteringen @Audited. Detta kompletterades även med 2 klasser för att ta hand om vem som gjort ändringarna. De 2 klasserna är LeaRevEntity.java och LeaRevListener.java. LeaRevEntity är objektet som tar hand om den information som ska sparas. LeaRevListener letar upp identiteten på den som gör en viss ändring.

Till detta gjordes en sida [Figur 3] som kan bli ärvd av andra sidor för att visa de ändringar som blivit gjort på ett objekt.

@@lea.log@@

@@lea.changedate@@	@@lea.changedate@@	@@lea.changedate@@	@@lea.changedate@@	@@lea.changedate@@
REV-1 BY ldc-jha	1	May 26, 2010 8:34:05 PM	130.235.40.187	getId::{11149} getComment::{HP 8510w} getDomainName::{null} getNet::{130.235.40.0/24} getDhcpgroup::{null} getHostNames::{{[EGAWS2359.ldc.lu.se]}}getFullDescription::{130.235.40.187} isRestricted::{false} getIppool::{null} getIpAddress::{130.235.40.187} getIpBinary::{10000010111010110010100010111011} getLastUsedByMac::{null} getLastUsedTime::{null} getLastUsedSwitchPort::{null} getReverseHostName::{jocke2.ldc.lu.se} getLucatId::{ldc-jha} getMacAddresses::{{[00:17:A4:EA:57:52, 00:12:E8:CE:91:E5]}} getLastChangedate::{2010-03-30 10:27:41.497} getLastChangedby::{Joachim Andersson (ldc-jha)} getLucatFullName::{null}
ldc-jha	1583	Jun 8, 2010 1:57:34 PM	130.235.40.187	getLucatFullName::{Joachim Andersson}
ldc-jha	2166	Jun 15, 2010 12:46:47 PM	130.235.40.187	getComment::{HP 8510w Jocke}
ldc-jha	2167	Jun 15, 2010 12:47:00 PM	130.235.40.187	getComment::{HP 8510w}
ldc-jha	5440	Jul 5, 2010 10:09:10 AM	130.235.40.187	getMacAddresses::{{[00:12:E8:CE:91:E5]}}
ldc-jha	5441	Jul 5, 2010 10:09:11 AM	130.235.40.187	getMacAddresses::{{[]}}
ldc-jha	5443	Jul 5, 2010 10:09:23 AM	130.235.40.187	getHostNames::{{[]}}getReverseHostName::{jocke.ldc.lu.se}
ldc-jha	5448	Jul 5, 2010 10:12:08 AM	130.235.40.187	getMacAddresses::{{[18:A9:05:C3:06:57]}}
ldc-jha	5450	Jul 5, 2010 10:12:31 AM	130.235.40.187	getComment::{Ny Dator(Jocke)} getHostNames::{{[EGAWS3330.ldc.lu.se]}}
	Last rev:10472	Sep 9, 2010 11:27:57 AM	130.235.40.187	

Figur 3: Skärmdump av Logg utskriften

2.4.1.7 Sökning

När det kommer till sökningen implementerades 2 olika alternativ. Alternativ 1 använder sig av Hibernat Search ramverket som är ett fulltextsökningsverktyg. Fördelen med detta är att det är lätt att använda sig av, man definierar i de olika entiteterna vilka attribut som ska indexeras och då vara sökbara. Detta går väldigt snabbt att söka via Hibernate Search. Problemet med detta är att det inte går att begränsa sökningen så att man bara kan söka på de objekten(Nät/Domäner/IP-adresser/Cnamn/Hostname) som man har rättigheter till.

För att komma runt detta gjordes även en sökfunktion som inte använder sig av Hibernate Search utan där det skrevs sökfunktioner manuellt. Alla objekt är kopplade till en Customer antingen direkt eller via ett annat objekt och på detta sätt kan det avgränsas så att man inte kan söka fram objekt som inte tillhör dem man har rättigheter till.

2.4.2 Implementation av databasen

I implementationen av databasen så finns det en javaklass för var och en av tabellerna som finns i databasen. Dessa javaklasser heter entitetsobjekt och definieras med att man lägger till en annotation i början av klassen @Entity.

Också alla relationer definieras i dessa objekt. Detta gör man ganska enkelt via Hibernate. Med denna rad ovanför en metod:

```
@OneToMany(cascade = CascadeType.REFRESH,fetch = FetchType.LAZY, mappedBy = "customer")
public List<Net> getNets() {
		return this.nets;
}
```

I varje entitetsklass kan man lägga till kontroller eller restriktioner. T.ex. som @MaxLength eller @NotNull. Med hjälp av dessa kontrollfälten kan inte entiteten sparas om inte dessa uppfylls.

2.4.3 Validatorer

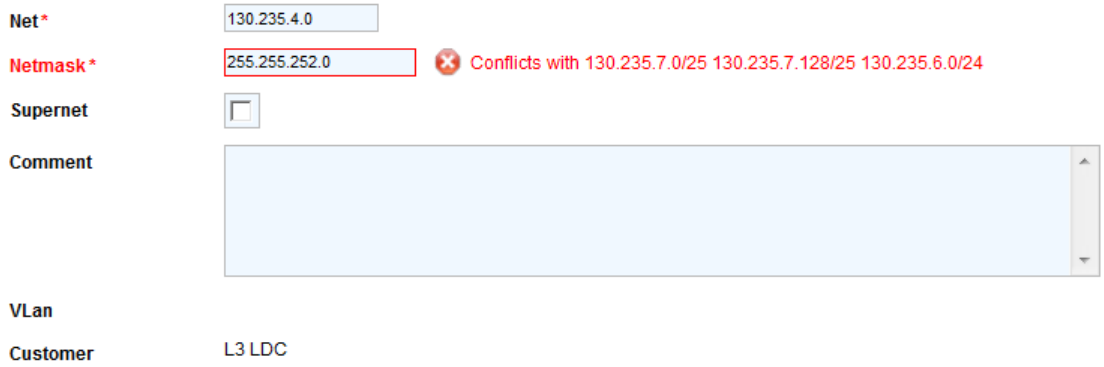
Alla sidor med värden som går att ändra finns det en validator kopplat till.
I projektet används två typer av validatorer. Det finns de som är kopplat direkt tillbaka mot databasen. Dessa specificerar man direkt på objektet som definierar databasen d.v.s. entiterna. De vanligast som används är @NotNull, @Length.

Den andra typen av validatorer är de som är inte är kopplade mot databasen. Denna typ av validator används bara när man ändrar något via GUI. De som finns är när en användare skriver in ett cnamn och vilket mål cnamet ska peka på, t.ex. att nettools.lu.se ska peka på a0038.srv.lu.se så slår den validatorn upp mot DNS-servern och kollar så att a0038.srv.lu.se är en giltig adress.

Några exempel på validatorer som är gjorda:

- Kontroll av Macadresser så det inte finns några dubbletter på ett objekt.
- Kontroll av format på HostNamn så att de innehåller rätt typ av bokstäver och att värdet börjar på en bokstav. Även kontroll att det inte finns något Cnamn som heter samma.
- Kontroll av Cnamn. Samma kontroll av hostnamn dock att den kollar så det inte finns något Cnamn med samma namn som det finns hostnamn.

- Kontroll av Cnamns mål, så att det är en giltig adress.
- Kontroll av att ett Nät är giltigt baserat på vilken nätmask den har. T.ex. att 130.235.40/0 är ett giltigt nät medan 130.235.40.128/24 inte är det. Även kontroll att nätet inte kommer överlappa ett annat nät. [Figur 4]



Figur 4: Exempel på hur en validator ser ut när man skrivit in fel värde.

2.4.4 Generering av DNS och DHCP Filer

Varje gång en ändring görs i någon av tabellerna ska det genereras nya DNS och DHCP filer för de motsvarande domänerna och näten. Detta är gjort via Hibernate, man kan lägga till Lyssnare som har koll på när ett objekt uppdateras, tas bort eller läggs till. Varje gång ett objekt uppdateras så hamnar de motsvarande näten och domänerna i en lista som kontrolleras var 5e minut om vilka filer som ska genereras. När filerna är skapade så förs de över till respektive DNS och DHCP server. Så det längsta det kan ta för en uppdatering att slå igenom är 5 minuter.

2.4.4.1 DHCP

Det är uppdelat så att varje nät har en fil t.ex. 130.235.40.0/24 nätet ska ha en fil som heter 130.235.40.0. Var gång en sådan här fil genereras så kontrolleras den mot programmet dhcpd innan filen kommer att föras över till DHCP servern. Om det är några fel i filen hamnar på felloggen.

2.4.4.2 DNS

DNSen består av 2 typer av filer som ska genereras. Den första typen är framlänges filer, som används att översätta t.ex. www.lu.se till 130.235.62.56. Dessa filer är uppdelade per Domän så t.ex. lu.se filen kommer ha en egen fil som heter lu.se

Den andra typen av filer som kommer att generas här är baklängesfilerna, de som slår upp att 130.235.62.56 går mot a0057.srv.lu.se. Dessa är baserade per hela /24 nät, dvs varje fil innehåller 256 adresser, och är baserade per nät. T.ex. filen för 130.235.40.0/24 kommer heta rev-130.235.40

Innan dessa filer förs över till DNSen kommer dessa att kontrolleras mot programmet named-checkzone, om där e några problem med filerna förs det inte över och felen hamnar på fel loggen.

2.4.5 Cronjobb

Varje natt kommer det köras några jobb som uppdaterar och kontrollerar en del saker i systemet.

Varje natt genereras alla DNS och DHCP filer om. Vissa fält innehåller ett ID som finns i Ldap, varje natt uppdateras all information t.ex. namn då vissa personer ibland byter namn. Det finns ett index för sökning via fulltextsökning som uppdateras. Det den gör är att den söker igenom alla objekt i databasen och gör en sökbar fil av detta. Det har även gjorts en kontroll för Cnamn, som kollar så att det som Cnamnet pekar på är giltigt.

Om något i dessa Cronjobben fallerar så kommer det att hamna på felloggen som den administrativa personalen på LDC kan se.

Vid utveckling av dessa jobb används ett system som kommer med Seam, och det heter Quartz. Vid uppstart av programmet registrerar man vilka funktioner som ska köras och med vilket intervall och vilken starttid.

2.5 Problem

2.5.1 Prestandaproblem vid visning av stora listor.

Vid visning av stora listor fanns problem med att det tog väldigt lång tid att generera sidorna. Först undersöktes utläsningen från databasen, men när tiderna kontrollerades var dessa relativt små och låg runt 20-30ms. Problemet var att det som hade missade var när det gällde visningen på sidan så gjorde den en SQL-sats för varje länkning av varje element, d.v.s. om en rad i visningen var länkat till t.ex. Nät och Customer så gjorde den 2 extra SQL för detta och när man ska visa 1000 rader blir detta ganska tidskrävande. Lösningen på detta var att skapa ett objekt för visning, och göra en SQL-sats som läser ut de elementen som behövs med en hjälp av en SQL-sats. Med hjälp av detta blev tiden väldigt långt, dock inte tillräckligt.

Vid anvädning oss av List objektet så anropades en funktion för få fram detta på Faclet sidan. Detta gjorde att varje rad anropade denna funktion och detta blev tidskrävande. Det som gjordes var att istället för anropa funktionen som byttes det till användning av Seams inbyggda annotation som heter @Datamodel. Vid användning av denna så läggs listan av objekt som en

annotering och man slipper anropa funktionen hela tiden utan kan använda sig av list objektet direkt.

Från att en sida tog ca 10 sekunder att visa ökades prestandan så att det nu tar ca 0.5 sekunder för en sida med 1000 objekt att visas.

2.5.2 Prestandaproblem vid inläsning eller ändring av listor.

Det var problem när man skulle läsa in stora listor av bulkändring eller ändring av stora listor. Det tog väldigt lång tid, så lång tid och så mycket processorkraft att systemet blev slött och tom kraschade. Vid utveckling av vissa funktioner fanns en timeout på 300 000 sekunder, denna ändras till 30 sekunder så att om något spårar ur så ska funktionen avbrytas. Detta var dock inte problemet som fanns, utan mer en säkerhetsåtgärd som gjorde så att servern inte skulle krascha. Problemet som var vid ändring av många objekt var att det användes de inbyggda funktionerna för detta. Alla objekt man använder sparas i det som kallas EntityManagern tills man tömmer denna och det görs på många olika sätt. Automatiskt när man avslutar en konversation i systemet eller om man gör det manuellt. När en ändring på en stor lista gjordes så sparades dessa objekt i EntityManagern och varje gång man uppdaterade ett objekt så sparades alla de objekten som funnits med i listan sedan innan. Det som gjordes var att inte använda det inbyggda utan uppdaterade alla objekt i listan och sen en spara, istället för vart och ett av objekten. Detta gjorde att en funktion som var för att uppdatera 100 objekt gick från att ta ca 10 minuter till det mer rimliga 20 millisekunder.

3 Sammanfattning och slutsatser

Det största målet för LDC med utvecklingen av denna applikation var att man skulle få möjlighet till intern vidareutveckling och få bort det personberoende som fanns till Banhof.

Tack vare att utvecklingen för NetTools skett i ramverket Seam så har alla utvecklare på LDC möjlighet att göra ändringar i systemet, då Seam är det som ska användas för utveckling av alla javaapplikationer på LDC. Detta kommer göra det lättare vid utveckling av nya applikationer på LDC, personalen på LDC kommer lättare kunna sätta sig in i andras program då de bygger på samma ramverk och man kan då också lättare komma runt rådande problem med att applikationen är personberoende.

Vid utveckling av NetTools är det även gjort en del funktioner och hantering av språkmenyer etc. som kommer ingå i det ramverk LDC kommer använda för framtida utveckling av applikationer i Seam.

Det har gjorts en utvärdering av den gamla databasen, och man kom då fram till att den inte höll de mått som behövs för utveckling av applikationen. I projekt har då designats en ny databasmodell. På grund av den nya databasmodellen kommer nya utvecklare att lättare kunna följa flödet i applikationen, och få det lättare att göra ändringar. Det är lättare och se vilka attribut som tillhör ett visst objekt. Den nya databasmodellen stöder vårt ramverk som används för utvecklingen bättre än vad den gamla modellen gjorde.

Ett användarvänligt gränssnitt med möjlighet för att användarna att uppdatera, ta bort och lägga till objekt är utvecklat. Man kan även se loggar och de genererade filerna som ska överföras till DNS och DHCPen.

Varje natt genereras alla DNS- och DHCP-filer om för att kunna överföras till respektive DNS och DHCP, det finns även ett jobb som ligger och körs som kontrollerar om något objekt blivit uppdaterat och genererar ny fil till respektive objekt som blivit ändrat.

När man i framtiden vill lägga till möjligheten att hantera switchar så kan man enkelt lägga till de tabellerna utan att påverka existerande struktur och koppla dem till en kund för uppdelning om vem som ska ha tillgång till vad. Det ska även vara möjligt att lägga till ytterligare funktionalitet utan problem när så önskas.

Efter test av testare och piloter har styrgruppen godkänt att applikationen möter de krav som har ställts på den.

4 Källor

4.1 Internet

[1] http://seamframework.org/ [2010-05-20]
[2] http://docs.jboss.org/seam/2.2.1.CR1/reference/en-US/html_single/ [2010-07-22]
[3] http://www.redhat.com/ [2010-07-22]
[4] http://www.hibernate.org/[2010-07-22]
[5] http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html [2010-07-22]
[6] http://www.jboss.org/richfaces [2010-06-10]
[7] https://facelets.dev.java.net/ [2010-06-10]
[8] http://www.quartz-scheduler.org/ [2010-08-01]
[9] http://www.jasig.org/ [2010-01-10]

4.2 Litteratur

12 Beginning JBoss® Seam: From Novice to Professional [978-1590597927]
13 Beginning JSF 2 APIs and JBoss Seam [978-1430219224]
14 Seam 2.X Web Development [978-1847195920]

5 Bilaga 1

Ordlista över termer och förkortningar

Här är en samling av alla termer och förkortningar som används i detta examensarbete.

Banhof
Är namnet på det företag där LDC köpte in deras första system för att kunder skulle kunna hantera DNS och DHCP. Systemet som köptes in fick namnet LUDD.

Cnamn
Även kallad Alias. Är en typ av objekt som används i DNSen för som är alias för ett hostnamn(Anamn).

CAS
Är en förkortning på Central Authentication Service och är en tjänst för hantering av autentisering. I Lunds fall körs autentisering mot Ldap.

CFL
Är en förkortning på Centralt Funktionsansvar för Lokala nät. Detta är ett projekt som har pågått på LDC för att få ett gemensamt Nät och drift på Lunds Universitet. Utvecklingen av NetTools är en del av detta projekt.

Cronjobb
Är namn på jobb, funktioner, program som körs med jämna intervaller.

DHCP
Är en förkortning på Dynamic Host Configuration Protocol. DHCP används vid all hantering av IP-adresser t.ex. så som utdelning av en IP-adress till en viss dator.

DNS
Är förkortning på Domain Name System, som används vid översättning av namn till en IP-adress eller från en IP-adress till ett namn.

Hibernate
Är ett ramverk för hantering av databasfrågor, sökning, versionshantering och uppbyggnad av entiteter.

JSF
Java Server Faces (JSF) är ett ramverk för att bygga webbapplikationer. JSF används för att bygga upp alla presentationssidor för applikationen.

Ldap
Är en förkortning för Lightweight Directory Access Protocol. Detta är den tjänsten LDC använder sig av för alla katalogdata om de anställda och studenter på Lunds Universitet

LUDD
Är namnet på det System som köptes in från Bahnhof och står för Lunds Universitets DNS och DHCP.

NetTools
Detta är namnet på systemet som utvecklas i detta examensarbete.

Opensource
Är namn på all öppen programvara där källkoden är tillgänglig att använda, läsa, modifiera och vidaredistribuera för den som vill.

PTR
Är namnet på det inläggen i DNSen som användes för bakåt uppslagning. D.v.s. översättning av 130.235.40.187 till jocke.ldc.lu.se.

Resourcebundel
Innehåller språk specifika objekt i java. Används oftast när man har översättning av en applikation.

Seam
Seam är en opensource plattform för utveckling av webbapplikationer.

VLAN
Är en förkortning på Virtual LAN som är en teknik för att ge användargrupper tillgång till ”egna” fasta nät, fast dessa är enbart virtuella inom samma fysiska nät.