Datavetenskap

Christian Ekström och Per Rydberg

Decentraliserad administration av gästkonton vid Karlstads universitet

Decentralized Management of Guest Accounts at Karlstad University

C-uppsats 15 hp
Datavetenskap

Datum/Termin: 10-06-11
Handledare: Stefan Lindskog
Examinator: Martin Blom
Löpnummer: C2010:09

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

Decentraliserad administration av gästkonton vid Karlstads universitet

Christian Ekström och Per Rydberg

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är vårt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Christian Ekström

Per Rydberg

Godkänd, 2010-06-11

Handledare: Stefan Lindskog

Examinator: Martin Blom

Sammanfattning

Denna rapport beskriver ett examensarbete som gjordes åt IT-avdelningen vid Karlstads universitet. Målet med examensarbetet var att skapa ett webbaserat system för hantering av gästkonton i ett Active Directory. Ett gästkonto ger tillgång till universitetets IT-miljö. Systemet ska vara tillgängligt för personalen på universitetet genom det befintliga inloggningssystemet CAS. Examensarbetet bestod av att ta fram en detaljerad kravspecifikation för systemet samt att implementera en prototyp. Prototypen består av två delar. En användardel och en administratörsdel. Personalen på universitetet kommer att via CAS logga in på användardelen där de kan hantera gästkonton. De finns funktionalitet för att skapa, ändra och ta bort gästkonton. Alla ändringar som sker skrivs till en logg som sparas i en databas. Personal vid IT-avdelningen loggar också in via CAS och skickas då till administratörsdelen. Där kan de se hur systemet har använts över tid med hjälp av den logg som finns i systemets databas. Det finns även funktionalitet för administratören att lista samtliga aktiva gästkonton och att avaktivera ett eller flera av dessa. Resultatet av examensarbetet är en väl fungerande prototyp för att hantera konton i ett AD.

Decentralized Management of Guest Accounts at Karlstad University

This report describes a bachelor project which was done at the IT department at Karlstad University. The aim of this project was to create a web-based system for managing guest accounts in an Active Directory. A guest account provides access to the university's IT environment. The system should be accessible to the staff at the university through the existing login system CAS. The project consisted of developing a detailed specification for the system and to implement a prototype. The prototype consists of two parts, a user side and administrator side. The staff at the university will via, the CAS system, logon to the user side in which they can manage guest accounts. The functionality is to create, modify, and delete guest accounts. All changes will be written to a log which consists of a database. Staff at the IT department also log on via the CAS system and is then sent to the administrator side where they can see how the system has been used over time using the log that is stored in the database. There is also functionality for the administrator to list all active guest accounts and to disable one or more of these. The result of the bachelor project is a well-functioning prototype for managing accounts in an AD.

Innehåll

Figurer

1 Introduktion

Denna rapport beskriver ett examensarbete på C-nivå i datavetenskap vid Karlstads universitet. Examensarbetet har gått ut på att skapa ett nytt webbaserat system för gästkontohanteringen på Karlstads universitet. Det på uppdrag från IT-avdelningen på Karlstads universitet. Det finns idag två olika system för hantering av gästkonton. Examensarbetet syftar till att skapa ett nytt system som ska ersätta de båda gamla systemen. Detta genom att skapa ett webbaserat system som ska decentralisera administrationen av gästkonton till universitetets IT-miljö. Hanteringen av gästkonton ska flyttas från IT-avdelningen till övrig personal på universitetet. För att göra detta skapade vi en prototyp av ett system med två olika delar. En användardel och en administratörsdel. I användardelen finns funktioner för att skapa, ändra och ta bort gästkonton. Alla förändringar som sker på ett gästkonto med de funktionerna sparas till en databas, så att det via administratörsdelen går att se hur systemet har använts. Administratörsdelen kan se dessa loggar i databasen samt lista samtliga aktiva gästkonton i systemet. Administratören har även möjlighet att inaktivera ett eller flera gästkonton i systemet. Sådana operationer skrivs också till databasen.

Rapporten är uppbyggd av sex kapitel som beskriver examensarbetet. I kapitel 2 finns en översikt av universitetets IT-miljö och dagens gästkontosystem. Där finns också en genomgång av de verktyg vi använt för att utveckla det nya systemet. I kapitel 3 finns analysen över de alternativ på implementering som vi kom fram till och den kravspecifikation som vi jobbade efter. I kapitel 3 beskrivs också designen av vårt system och en detaljerad genomgång av systemet presenteras. Testning av systemet skedde kontinuerligt under utvecklingen. Hur denna testning utfördes tas upp i kapitel 4. I kapitel 5 tar vi upp de problem som vi stött på under utvecklingen. De innefattar problem med hårdvara, testmiljön och implementationen. I detta kapitel finns också några alternativa implementationslösningar och framtida förbättringar av systemet. Slutligen sammanfattar vi våra erfarenheter av examensarbetet i kapitel 6. I bilaga A presenteras en översikt av samtliga filer och funktioner i systemet.

2 Bakgrund

I det här kapitlet kommer vi beskriva bakgrunden till examensarbetet och dess målsättning. De existerande systemen för administration av gästkonton presenteras samt en beskrivning av Karlstads universitets IT-miljö. De system och tekniker som används kommer att presenteras och beskrivas. Vi presenterar även de programspråk och verktyg som är aktuella för examensarbetet.

2.1 Översikt

Syfte med det här examensarbetet är att skapa en webbapplikation för hanteringen av gästkonton i Karlstads universitets IT-miljö. Universitetets IT-avdelning sköter idag all hantering rörande gästkonton. Gästkonton administreras idag med hjälp av två olika system. Dels med en webbapplikation som utvecklades av IT-avdelningen för drygt tio år sedan. Den applikationen används för att hantera gästkonton till det trådlösa nätverket. För att hantera de gästkonton som används i datorsalar och till de tunna klienterna används Active Directory Users and Computer (ADUC) [36]. Problemet med de existerande systemen är att det är bara IT-avdelningen som har åtkomst till och kan använda dem. Det gör att när personal på universitetet behöver gästkonton till exempelvis deltagare på en konferens, så måste IT-avdelningen kontaktas i förväg så att gästkonton kan skapas. Även om det finns ett system för hanteringen av gästkonton så tar gästkontohanteringen upp onödig tid för systemadministratörerna. Tid som kan användas till andra saker om övrig personal kunde hantera sina egna gästkonton utan att IT-avdelningen behöver vara inblandad. Målet med examensarbetet är därför att skapa en webbapplikation som ska ersätta de båda gamla systemen och då slå ihop hanteringen av gästkonton från två olika system till ett och samma, samt att ge övrig personal på universitetet möjlighet att själva hantera sina gästkonton med tillhörande dokumentation. Hantering av gästkonton innebär funktionalitet för skapa, ändra och radera konton. Med dokumentation menas

dokument, både digitala och papperskopior med kontouppgifter. Till övrig personal räknas alla anställda med ett personalkonto i IT-miljön. Med den nya webbapplikationen förändras IT-avdelningens arbete rörande gästkonton till en övervakande roll. Den nya webbapplikationen har funktioner som loggar alla kontoförändringar i systemet, när ett konto skapas, när ett konto ändras och så vidare. Dessa loggar kommer av IT-avdelningen utsedda administratörer ha tillgång till via ett administratörsgränssnitt i webbapplikationen. Anledningen till att alla förändringar av konton loggas är att det ska finnas spårbarhet i systemet. Det måste vara möjligt att spåra gästanvändare som missköter sig eller om personal använder systemet felaktigt.

2.1.1 Karlstads universitets IT-miljö

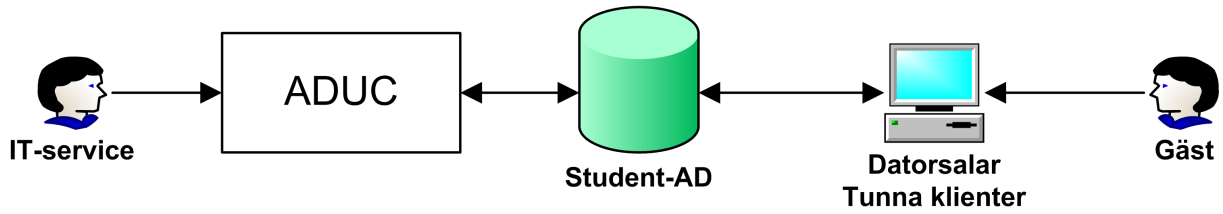
Den centrala funktionen för Karlstads universitets IT-avdelning är att hantera och ansvara för driften av verksamhetens IT-system. IT-avdelningen hanterar ungefär 3000 datorer och 30000 användarkonton i två komplexa IT-miljöer. En miljö för studenterna och en för personal. Alla studenter på Karlstads universitet har tillgång till ett användarkonto som ger åtkomst till flera tjänster och funktioner. Dessa konton lagras i ett Microsoft Active Directory (AD). Med kontot kan studenten logga in på de stationära datorerna i datorsalarna, logga in på de tunna klienter i biblioteket och i hus 21 samt ansluta till det trådlösa nätverket "KAU-STUDENT". Kontot har även ett tillhörande e-postkonto med adressen `användarnamn@student.kau.se`. `Användarnamn` kallas också för netID. Kontot används även för att logga in i tjänsten "Mina Sidor" där studenten kan se vilka kurser han/hon är registrerad på, registrera sig på nya kurser, anmäla/avanmäla sig till tentamen, se resultat från avslutade kurser och se totalt antal avklarade högskolepoäng.

Personalens konto ger tillgång till en mängd funktioner som exempelvis Kaucentral där det finns tjänster som kursadministration, bokningslistor och personalkatalog med mera. Personalen kan också sköta tentamensadministration, resultatrapportering och webbpublicering på `www.kau.se`. Även personalen har ett e-postkonto där netID är lika med använ-

darnamn på e-postkontot, så att personalens e-postadressen blir således `netID@kau.se`.

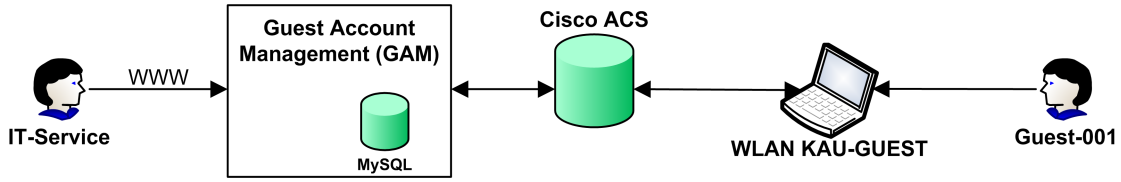
2.2 Existerade lösningar

Det finns idag två olika system för att hantera gästkonton på Karlstads universitet. Det finns ett system för hanteringen av de gästkonton som används till datorsalarna och till de tunna klienterna. Det andra är ett system för det trådlösa gästnätverket. Gästkonton för datorsalar och tunna klienter finns i samma AD som studenternas konton. För att hantera dessa konton används ADUC, se Figur 2.1.



Figur 2.1: Gästkonton datorsalar och tunna klienter.

För hanteringen av gästkonton till det trådlösa nätverket `KAU-GUEST` används ett webbaserat system. Det systemet utvecklades för fem år sedan, för att förenkla och snabba upp hanteringen av gästkonton. Utsedda samordnare skulle ha tillgång till systemet för att kunna skapa nya gästkonton utan att först ha kontaktat IT-avdelningen. Nu blev det tyvärr inte så, de personer som fick tillgång till systemet var personal på IT-avdelningen som behövde åtkomst för att kunna skapa gästkonton till övrig personal på universitetet. Sedan lanseringen av systemet har det används för att skapa i genomsnitt 1100 gästkonton per läsår. Webbsystemet är uppbyggt så att användaren loggar in via Central Authentication Service (CAS) och sedan skickas vidare till webbsystemet där flertalet funktioner finns tillgängliga. Systemet bygger på en Cisco Access Control Server (ACS) [15] tillsammans med en MySQL-databas. Allt som görs via hemsidan skrivs till MySQL-databasen. Där mellanlagras datan för att sedan skrivas till databasen i ACS med hjälp av ett schemalagt skript, se Figur 2.2.



Figur 2.2: Gästkonton för trådlösa nätverket KAU-GUEST.

2.3 System och tekniker

I detta delkapitel kommer vi gå igenom de system och tekniker som vi har använt vid utvecklingen av webbapplikationen.

2.3.1 Active Directory

AD [29] är Microsofts [12] katalogtjänst. AD ingår som en del i serverversionerna av Microsoft Windows sedan lanseringen år 2000. AD ger administratörer möjlighet att enkelt skapa policys för användare och grupper och distribuera program till flera datorer i en IT-miljö. AD lagrar information och inställningar i en central databas som kan distribueras till flera servrar i nätverket. Storleken på ett AD-nätverk kan variera från ett litet företagsnätverk med bara några få datorer till stora komplexa nätverk med tusentals datorer och användare som bygger på flera domäner och servrar. AD är en katalogtjänst som är uppbyggd av olika typer av objekt som har olika attribut. AD är en hierarkisk uppbyggd miljö med en så kallad skog längst upp. En skog kan i sin tur innehålla ett eller flera träd och träden i sin tur innehåller en eller flera domäner. Även domänerna kan ha subdomäner som i sin tur kan ha subdomäner och så vidare. Domäner och subdomänerna kan innehålla objekt. Ett objekt kan vara en dator, användare, en grupp, en delad mapp, en skrivare med mera. Även ett Organisation Unit (OU) [35] kan vara ett objekt. Ett OU eller en organisationsenhet är en typ av mapp som innehåller tidigare nämnda objekt. Ett OU kan även innehålla andra OU. Det är med hjälp av organisationsenheter som objekt struktureras upp i en domän eller subdomän.

2.3.2 Lightweight Directory Access Protocol

Lightweight Directory Access Protocol (LDAP) [21] är ett applikationsprotokoll som används för att kommunicera med en katalogtjänst över Transmission Control Protocol/Internet Protocol (TCP/IP) [8]. LDAP bygger på Directory Access Protocol (DAP) [9] som utvecklades under 1980-talet för att kommunicera med en X.500-katalogtjänst [9]. LDAP var i början ett mindre och lättare protokoll än DAP just för att förenkla kommunikation med en katalogserver. LDAP har dock i senare versioner byggts på så att det nu är lika stort som DAP. I version tre finns elva olika operationer inbyggda. De är operationer för att kunna ansluta till en katalogserver, söka, skapa, modifiera och ta bort objekt samt flera andra operationer. LDAP version tre är den senaste versionen av protokollet. Det blev standardiserad av Internet Engineering Task Force (IETF) [16] i december 1997.

2.3.3 Central Authentication Service

CAS [27] är ett webbaserat Single Sign-On-system (SSO) [22] med öppen källkod som används för olika typer av webbtjänster. CAS är utvecklat av Jasig [24]. Karlstads universitet använder CAS som inloggningssystem för ett flertal tjänster som ingår i universitetets IT-miljö och som studenter och personal har tillgång till. Med CAS behöver användaren bara logga in en gång och sedan sköter CAS åtkomsten till de tjänsterna användaren vill komma åt. Flera system använder samma inloggningsuppgifter som CAS (NetID och lösenord) även om de inte använder CAS-gränssnittet. Ett exempel är det trådlösa nätverket som finns på universitetet. När en användare vill logga in på en tjänst så skickas användare vidare till CAS där han/hon anger användarnamn och lösenord. CAS verifierar sedan användarnamn och lösenord mot katalogen med studenter och personal. Om autentiseringen lyckas skickas användaren tillbaka till applikationen med en så kallad biljett. Applikationen kontrollerar biljetten genom att skicka tillbaka den till CAS tillsammans med sin egen service-identifiering. CAS kontrollerar biljetten och skickar sedan information om användaren till applikationen.

2.3.4 MySQL

MySQL [19] är en relationsdatabashanterare. MySQL är en fri programvara med öppenkällkod under en GNU-licens. Det är en svensk mjukvara utvecklad av Michael Widenius och David Axmark. De grundade MySQL AB år 1995 och köptes upp av Sun Microsystem 2008. Namnet MySQL kommer av My, dotter till Michael Widenius och SQL som står för Structured Query Language [17]. SQL används för att lägga till, hämta, uppdatera och ta bort lagrad data. MySQL är ett klient-serversystem och kan installeras på ett stort antal plattformar. MySQL levereras med en kommandotolk för åtkomst till databaser. Det finns också ett flertal grafiska gränssnitt som gör hanteringen lättare, till exempel MySQL Workbench, som är Sun Microsystems egna, eller phpMyAdmin som är en tredjepartsapplikation.

2.4 Webbutveckling

I det här delkapitlet kommer de fyra program- och skriptspråk vi har använt för utvecklingen av webbapplikationen att presenteras.

2.4.1 HTML

HyperText Markup Language (HTML) [28] är ett uppmärkningsspråk som används för att formatera utseende på webbsidor. HTML bygger på Standard Generalized Markup Language (SGML) [14]. HTML använder så kallade ”taggar” för att definiera olika element i ett dokument. Ett exempel är $<p>$ taggen som börjar ett nytt stycke text och $</p>$ som avslutar stycket. Den HTML-versionen som används idag är version 4.01, som 1997 fastställdes av World Wide Web Consortium (W3C) [10], version 5 är under utveckling.

2.4.2 CSS

Cascading Style Sheets (CSS) [6] är ett stilspråk som beskriver utseendet på ett strukturerat dokument som t.ex. typsnitt, textstorlek, bakgrundsfärg och olika elements position. Detta för att anpassa dokumentet efter vad klienten har för dator, upplösning på skärmen, installerade teckensnitt med mera. Ett HTML-dokument har i grunden ingen struktur på utskriften. Det består bara av text utan information om hur utskriften ska formateras. Hur utskriften av ett HTML-dokument kan formateras kan bestämmas med en stilmall skriven i CSS. Idag använder i princip alla webbplatser CSS för att strukturera utseendet på webbplatsen.

2.4.3 PHP

PHP: Hypertext Preprocessor (PHP) [2] är ett skriptspråk baserat på öppen källkod som används vid utveckling av dynamiska webbsidor och webbapplikationer. Vid utveckling av webbsidor bäddar man ofta in PHP i HTML-kod. PHP-koden körs sedan på webbservern som returnerar HTML-kod. PHP finns tillgängligt för de flesta moderna webbservrar men finns även i en fristående version som kan användas på ett flertal operativsystem utan att en webbserver behöver vara installerad. PHP skapades ursprungligen 1995 av Rasmus Lerdorf och har kontinuerligt utvecklats. Den senaste versionen är 5.3.2 och version 6.0 är under utveckling.

2.4.4 JavaScript

JavaScript [20] är ett objektorienterat skriptspråk som framförallt används på klientsidan i en webbapplikation. Koden körs i webbläsarens JavaScript-motor. Vanligast är att JavaScript, precis som PHP, bäddas in i HTML-sidor. Vanliga användningsområden för JavaScript är att kontrollera HTML-formulär, bildspel och visa eller dölja olika element på en hemsida. JavaScript utvecklades av Netscape i mitten av 90-talet och släpptes för första gången i december 1995 med webbläsaren Netscape Navigator 2.0B3.

2.5 Kapitelsammanfattning

I det här kapitlet har vi presenterat syfte och mål med examensarbetet. Vi har presenterat det nuvarande webbsystemet för hanteringen av gästkonton samt kort redovisat Karlstads universitets IT-miljö. Examensarbetet syftar till att skapa en webbapplikation där anställda på Karlstads universitet kan logga in och hantera sina gästkonton utan att kontakta IT-avdelningen. Hanteringen innebär att personalen ska kunna skapa, ändra och radera gästkonton samt skapa tillhörande dokumentation. Dokumentationen är dokument med kontouppgifter i digital form samt papperskopior. I och med att anställda på universitetet själva kan hantera sina gästkonton så kommer universitetets IT-avdelning inte längre behöva lägga tid på hanteringen av gästkonton. Det gör att IT-avdelningen sparar tid och resurser. IT-avdelningen får istället en övervakande roll där systemadministratörer via administratörsgränssnitt i systemet kan se loggar över vad som har hänt. De system och verktyg som användes vid utvecklingen har presenterats och beskrivits.

3 Analys, design och implementation

I det här kapitlet kommer vi beskriva den utvecklade applikationen och dess design. Vi beskriver den analys som gjordes för att ta fram den detaljerade kravspecifikationen. Vi förklarar också den systemdesign vi valde att använda. Vi presenterar alla de funktioner som finns i systemet och ger en detaljerad beskrivning av både användar- och administratörsdelarna.

3.1 Analys

En del av examensarbetet gick ut på att ta fram en kravspecifikation för applikationen. Som utgångspunkt för denna fick vi en beskrivning av ett önskat system och en lista med tillgängliga verktyg av handledaren på IT-avdelningen. Vi diskuterade olika alternativ och lösningar, båda själva och med våra handledare på IT-avdelningen. Vi diskuterade flera

olika frågor för att kunna bestämma hur webbapplikationen skulle utformas.

Ska det finnas förskapade konton i AD eller ska konton skapas av användaren? Ska ändringarna skriva direkt till AD eller mellanlagras i en MySQL-databas? Ska det finnas en tidsgräns för hur länge ett konto kan aktiveras? Hur många konton ska en användare kunna skapa och ha aktiva samtidigt?

Det bestämdes att vi skulle använda förskapade konton i AD. Det ansågs vara säkrare att inte kunna skapa helt nya konton i AD, då dessa potentiellt kan få rättigheter som inte ska tillhöra ett gästkonto. Det ger också administratören möjlighet att på ett enkelt sätt begränsa antalet gästkonton som kan vara aktiva samtidigt.

Den stora frågan i förstudien och i framtagandet av en detaljerad kravspecifikation var huruvida vi skulle mellanlagra användarinformationen i en MySQL-databas eller om den ska skrivas direkt till AD. Det finns flera för- och nackdelar med båda metoderna. Att mellanlagra användarinformation ger följande fördelar. En användare kan välja tidsperiod när ett gästkonto ska vara aktivt. Det blir heller ingen begränsning i hur många konton som kan aktiveras vid ett och samma tillfälle.

Nackdelarna med mellanlagring är följande. Det är krångligare rent implementationsmässigt att mellanlagra användarinformation. Mellanlagring medför att flera småsaker som är lätta att göra om användarinformation skrivs direkt till AD blir svårare att göra. Det finns flera exempel på det. Ett exempel är att kontrollera att antal konton som ska aktiveras inte överskrider det antal som finns lediga och då går att aktivera. Ett annat exempel är att gästkonton inte kan aktiveras samma dag som de ska användas, utan aktiveringen sker vid en specifik tidpunkt. Det skulle till exempel kunna realiseras genom att det varje natt körs ett schemalagt skript som skriver över ändringarna till AD. Det medför även ett säkerhetsproblem då det skulle vara nödvändigt att lagra lösenorden i klartext i databasen för att kunna aktivera konton vid aktuell tidpunkt. Ett alternativt skulle vara att lösenorden krypteras i databasen för att sedan dekrypteras när konton ska aktiveras.

Att skriva användarinformation direkt till AD har flera fördelar. Ett konto kan användas

direkt efter aktiveringen. Det behövs ingen mellanlagring av lösenord. Det är lättare att kontrollera hur många konton som totalt kan aktiveras. En administratör får bättre översikt över belastningen på AD. Det är lättare att implementera, applikationen blir mindre komplex då det blir en komponent mindre.

Det finns självklart också nackdelar med att skriva användarinformation direkt till AD. Konton kan bara aktiveras från och med aktuellt datum. En annan nackdel är att antalet konton som kan aktiveras vid ett tillfälle är begränsat. Det på grund av att LDAP inte hinner utföra hur många operationer som helst på den maximala exekveringstid som PHP har förinställd. Att skriva till en MySQL-databas går betydligt fortare än att skriva till AD med LDAP. Därför är antalet konton som går att aktivera vid ett och samma tillfälle begränsat till 100 stycken.

Vi bestämde i samråd med handledare att vi skriver användarinformationen direkt till AD. Den maximala tiden för hur länge ett gästkonto ska vara aktivt sattes till 30 dagar efter begäran från handledare på IT-avdelningen.

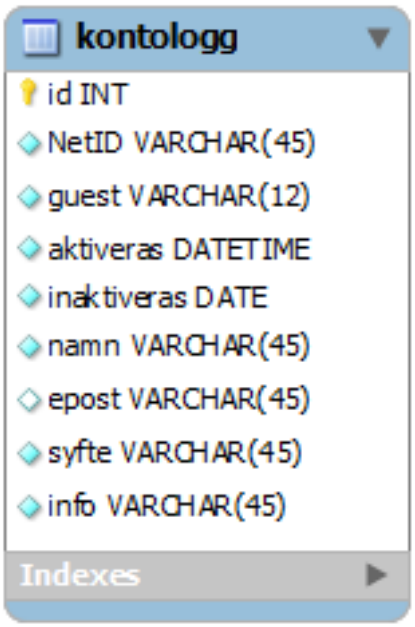
Efter dessa diskussioner togs en detaljerad kravspecifikation fram. När vi började implementera efter denna kravspecifikation, visade det sig att vi inte kunde uppfylla flera av de krav som fanns. Vi blev därför tvungna att revidera den. Det resulterade i en uppdaterad kravspecifikation som kunde implementeras. Kravspecifikationen finns i efterföljande delkapitel.

3.1.1 Kravspecifikation

Systemet är en webbapplikation. Personalen på Karlstads universitet loggar in till applikationen genom det befintliga inloggningssystemet CAS. Applikationen består av ett AD med 1000 fördefinierade konton, kontonamnen är *guest-xxx* där *xxx* är ett löpnummer. En MySQL-databas används för att logga dessa händelser i applikationen för att få spårbarhet. Med en händelse menas att ett konto aktiveras, avaktiveras eller ändras i AD.

Databasen innehåller en tabell med fälten `id, NetID, guest, aktiveras, inaktiveras,`

`namn`, `epost`, `syfte` och `info`, se Figur 3.1.



Figur 3.1: Struktur på loggdatabasen.

- `Id` - id används som primärnyckel.
- `netID` - netID på den person som hanterade kontot.
- `guest` - kontonamnet, *guest-xxx*.
- `aktiveras` - datum och klockslag för händelsen.
- `inaktiveras` - det datum som gästkontot avaktiveras.
- `namn` - namnet på gästanvändare av kontot.
- `epost` - den angivna e-postadressen till en gästanvändare.
- `syfte` - det angivna syftet med ett konto.

- `info` - vad som gjordes med gästkontot, aktiveras, ändrades, inaktiverades.

Det finns ett flertal funktioner för användare och administratörer i systemet. Funktioner i systemet för en inloggad användare ska vara:

- Aktivera nya gästkonton.
- Inaktivera egna konton.
- Lista egna konton som är aktiva.
- Ändra de egna konton som är aktiva.
- Generera Portable Document Format (PDF)-filer [26] med kontouppgifter.

Antalet konton som maximalt kan aktiveras är 100 per omgång. Det finns ingen begränsning på antalet omgångar mer än att de 1000 fördefinierade kontona tar slut efter tio maximala omgångar. Med egna konton menas de konton som en specifik användare har aktiverat. Användarna ska kunna ändra informationen på sina konton efter att de har aktiverats.

PDF-filerna som genereras är en sammanställning av alla konton från ett aktiverings- eller ändringstillfälle och kontoinformation om enskilda konton.

För att aktivera ett eller flera konton ska användaren ange följande information.

- Syfte med kontot, som exempelvis kan vara konferens.
- Giltighetstid, kontot aktiveras från och med dagens datum till det angivna datumet.
- Gästens namn eller motsvarade referens.
- E-postadress till gästen. E-postadress är inte obligatoriskt, men om det finns angivet skickas enskilda kontouppgifter till angiven adress.

Ovannämnda uppgifter skrivs som attribut till ett konto i AD, även netID på den person som har hanterat kontot skrivs till ett kontoattribut. Detta för att kunna identifiera vem som hanterat vilka konton.

När ett konto aktiveras slumpgenereras ett lösenord som sedan skrivs till det aktuella kontot i AD. Lösenordet består av tio tecken fyra versaler, fyra gemener och två siffror. Det ska gå att återställa lösenordet via hemsidan, för ett enskilt konto eller flera.

En i systemet inloggad administratör ska kunna:

- Lista alla aktiva konton.
- Avaktivera enskilda eller alla konton.
- Se loggen med användar- och kontohistorik.

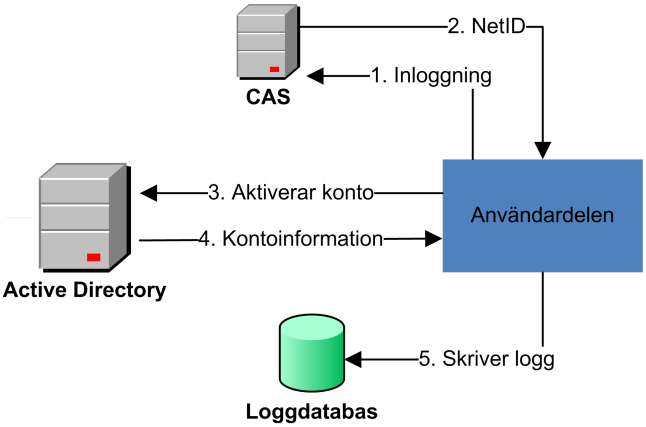
Användarhistorik innebär att ett netID anges och alla konton som har hanterats av det angiva netID visas. Kontohistorik innebär att ett kontonamn anges och alla poster i databasen där kontot förekommer listas. Den information som listas är alla fält från tabellen i databasen.

3.2 Övergripande design

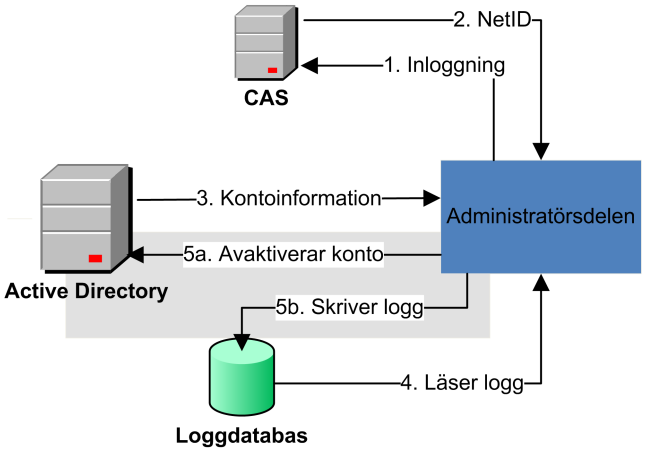
När den omarbetade kravspecifikationen blivit godkänd började vi att designa systemet. Utseendemässigt försökte vi efterlikna Karlstads universitets nuvarande webbplats. För användardelen av systemet skrev vi funktioner för att läsa och skriva till AD samt att skriva till databasen, se Figur 3.2.

Administratörsdelen av systemet har funktioner för att läsa och skriva till AD samt att läsa och skriva till databasen, se Figur 3.3. För inloggning mot CAS använde vi existerande kod som vi fick tillgång till via IT-avdelningen. Denna kod används också då vi hämtar NetID från CAS.

Det implementerade systemet består av fem olika delar:



Figur 3.2: Användardelen av gästkontohanteringen vid aktivering av konton.



Figur 3.3: Administratörsdelen av gästkontohanteringen.

- CAS
- AD
- Användardel
- Administratörsdel
- Loggdatabasen

CAS är inloggningssystemet som används för att logga in på de två olika användardelarna av systemet. Funktionen hos AD beskrivs i delkapitel 2.3 Användarsidan är den

delen av systemet som vanliga användare skickas till efter att de har loggat in via CAS. Administratörssidan är den delen av systemet som administratören skickas till efter inloggning via CAS. Det är bara personal på universitetet som tillåts logga in i systemet. CAS har i grunden inget sätt att särskilja studenter och personal. För att veta om det är anställd eller student görs en förfrågan (efter att inloggningen har skett) till en tjänst på en av IT-avdelningens servrar. Tjänsten svarar på om den inloggade personen är en student eller anställd. Om det är en student som försökt logga in, så loggas personen ut och skickas till CAS-startsida.

Användardelen av systemet ger användaren tillgång till flera funktioner. Användaren har efter inloggning tillgång till följande funktioner:

- Skapa nya gästkonton.
- Lista aktiva gästkonton som användaren har skapat.
- Ändra ett enskilt gästkonto.
- Inaktivera ett enskilt gästkonto i förtid.
- Återställa alla sina aktiva kontons lösenord.
- Inaktivera alla sina aktiva konton.

Administratörsdelen av systemet ger administratören tillgång till flera övervakningsverktyg och loggar över vad som skett i systemet. De funktioner administratören har tillgång till är följande:

- Se samtliga aktiva gästkonton.
- Inaktivera samtliga aktiva gästkonton.
- Se och söka i en logg över vad en specifik användare aktiverat, ändrat och inaktiverat för gästkonton.

- Se och söka i en logg över hur ett specifikt gästkonto har aktiverats, ändrats och inaktiverats.

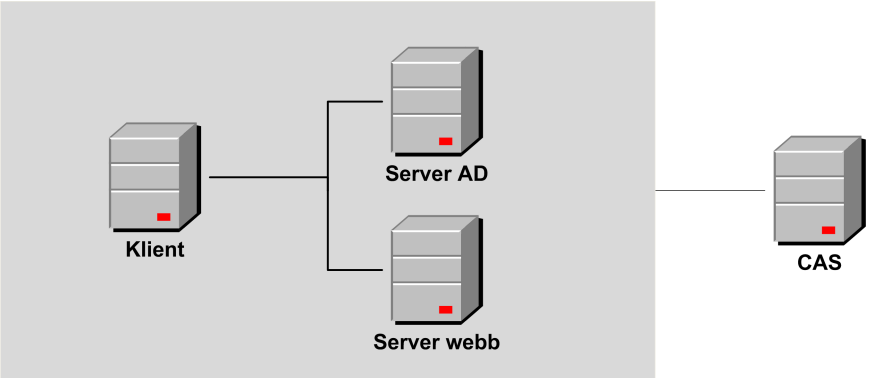
För att implementera dessa delar satte vi upp en testmiljö för vår applikation, se avsnitt 3.3.1. I kommande delkapitel ges en mer detaljerad beskrivning av systemet där funktionerna förklaras mer ingående.

3.3 Detaljerad beskrivning av systemet

I det här delkapitelet presenterar vi den testmiljö vi har använt under examensarbetet och hur den är uppbyggd. Vi ger en detaljerad beskrivning av användardelen och administratörsdelen. Vi går igenom vad som händer vid de olika funktionsanropen. I användardelen är dessa funktioner att skapa och hantera konton. I administratörsdelen är det avaktivering av konton och åtkomst till databasen. I bilaga A finns en mer detaljerad beskrivning av de filer och funktioner som nämns i 3.3.2 och 3.3.3

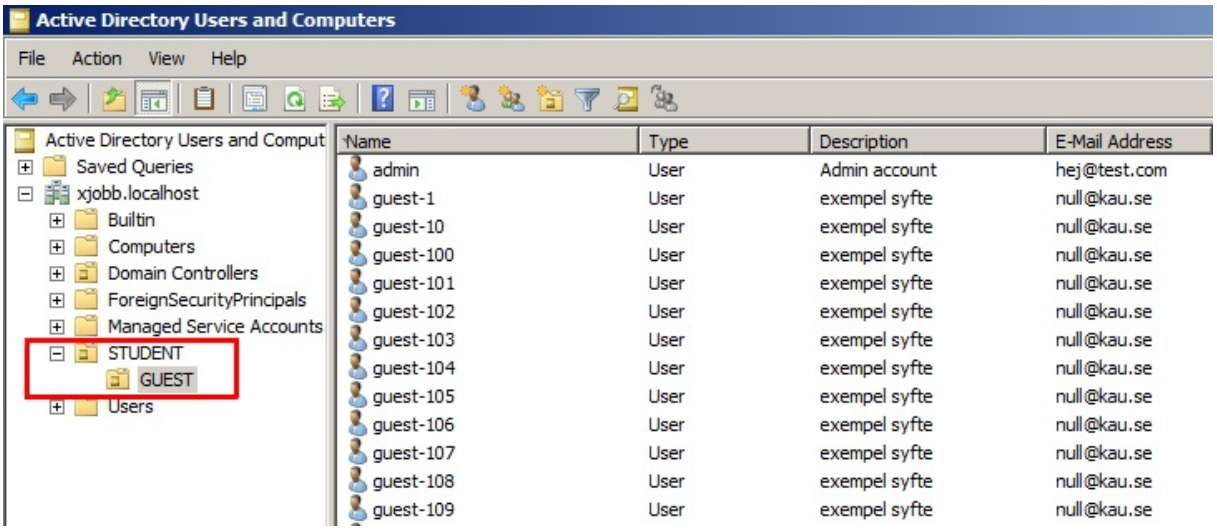
3.3.1 Testmiljö

Det första steget innan implementationen var att sätta upp en testmiljö för vår applikation med hjälp av programmet Virtualbox [25]. Testmiljön bestod av tre virtuella datorer, se Figur 3.4. En dator med Windows XP som operativsystem som agerar klientdator i det



Figur 3.4: Testmiljö.

virtuella nätverket och två stycken virtuella servrar med Windows 2008 Server Standard Edition som operativsystem. På den ena servern installerades ett AD. AD-strukturen var mycket enkel. Den bestod av två stycken OU, ”STUDENT” och ”GUEST”. Där ”GUEST” ligger under ”STUDENT”, se Figur 3.5. ”GUEST” innehåller de förskapade gästkontona



Figur 3.5: Strukturen i Active Directory.

samt ett administratörskonto för aktuellt OU. Administratörskontot har bara rättigheter i OU ”GUEST” och kan bara ändra på redan befintliga konton. Detta för att få så hög säkerhet i AD som möjligt. Detta kallas i litteraturen ”Need-to-know” [32]. Det innebär att det finns tillräckligt med information för att utföra en uppgift men inte mer än så.

På den andra virtuella servern installerades ett programpaket som heter WAMP-server [7] som innehåller, webbservern Apache [4], relationsdatabasen MySQL och en PHP-installation. Detta för att på ett enkelt sätt skapa en komplett webbserver med alla de delkomponenter som behövdes för testmiljön. Den virtuella klientdatorn anslöts till test-miljöns domän och användes för att kontrollera att det går att logga in med de gästkonton som har blivit aktiverade via webbapplikationen.

3.3.2 Användardel

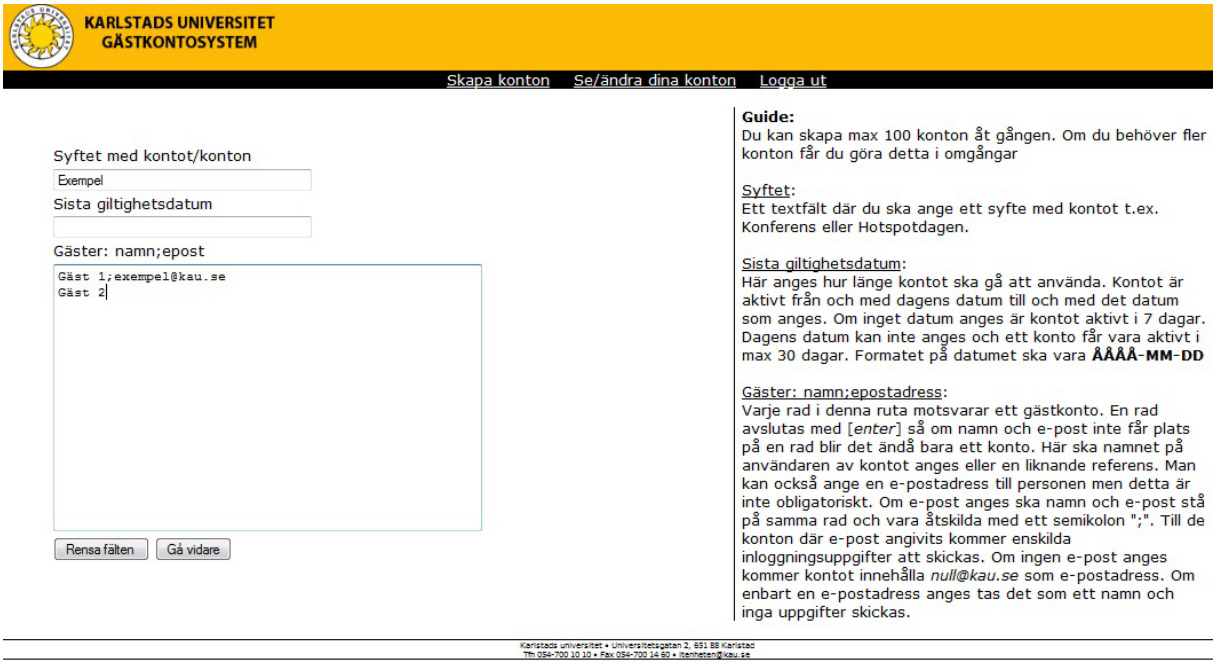
Efter lyckad inloggning via CAS ser användaren en sida med information om vad systemet är till för och antalet lediga konton, se Figur 3.6. Konton som är lediga är de konton som ännu inte är aktiverade i AD. Informationen om antal lediga konton hämtas från AD med en funktion som kopplas till ett administratörskonto i AD, och därefter räknas antalet konton som inte är aktiverade. Uppkopplingen mot administratörskontot i AD används sedan till all kommunikation mot AD. Här finns också länkar för att komma åt funktionerna i systemet. Dessa länkar visas på samtliga sidor som användaren kommer till så de går att komma åt funktionerna och att logga ut från CAS oavsett vad som tidigare gjorts. De funktioner som finns tillgängliga är:

- Aktivera konton.
- Se de konton som har aktiverats tidigare och ändra attribut på dessa.
- Logga ut från systemet.

Funktionerna för att se och ändra konton visar bara de konton som den inloggade användaren har aktiverat.

Figur 3.6: Startsida för en användare.

Skapa konton När en användare vill aktivera ett antal konton så ska ett HTML-formulär med två textfält, en textarea och två knappar fyllas i, se Figur 3.7. I textfälten anges



Figur 3.7: Skapa konto.

ett syfte med kontot och ett avaktiveringsdatum. I textarean anges namnen på gästerna och eventuellt en e-postadress. Syftet med ett konto kan till exempel vara en konferens. Konton aktiveras från och med dagens datum till och med det datum som angivits som avaktiveringsdatum. Den maximala aktiveringstiden för ett konto är 30 dagar. Om inget datum anges så blir kontona aktiverade i sju dagar. Namnen på gästerna kan vara faktiska namn eller en referens till en gäst om namnet inte är känt. I de fall som en e-postadress har angetts ska denna skrivas på samma rad som gästnamnet åtskilt av ett skiljetecken. Skiljetecknet som används är ett semikolon. En av knapparna rensar alla fälten och den andra tar användaren vidare.

När användaren går vidare kontrolleras all information från fälten. Syftet kontrolleras så att det inte är tomt och eventuella specialtecken tas bort av säkerhetsskäl. Ett exempel på vad som kan hända om specialtecken tillåts är en så kallad SQL-injektion [18]. En SQL-

injektion utnyttjar ett säkerhetsproblem som har att göra med hanteringen av indata till vissa program som arbetar mot en databas. Om ett datum är angivet kontrolleras att rätt format har använts. Detta görs för att det ska gå att göra om till den tidstämpel som AD kräver. Datumet kontrolleras också så att det inte redan passerats, att månad och dag är giltiga och att aktiveringstiden inte överstiger 30 dagar. Informationen från textarean delas först in i rader och om det finns tomma rader plockas dessa bort. Efter detta delas varje rad in i namn och e-post med hjälp av skiljetecknet. I både namn och e-post tas de flesta specialtecken bort av samma skäl som ovan. De specialtecken som tillåts är snabel-a, bindestreck, understreck och punkt då dessa är tillåtna tecken i en e-postadress. Antalet konton som ska aktiveras kontrolleras också då max 100 konton kan aktiveras på en och samma gång. Denna gräns har införts för att systemet inte ska avbryta exekveringen på grund av att för lång tid har gått sedan exekveringen började. Detta beror på att PHP:s maximala exekveringstid är satt till 30 sekunder. Om något av informationen är felaktig skickas användaren tillbaka till formuläret och ett felmeddelande skrivs ut.

När användaren går vidare och allt är korrekt så anropas två funktioner. En funktion genererar ett slumpmässigt tio tecken långt lösenord som består av fyra versaler, fyra gemener och två siffror. Detta uppfyller kraven på komplexiteten för ett lösenord enligt kravspecifikationen. Vissa bokstäver och siffror som kan misstolkas är borttagna som till exempel siffran 1 och bokstaven I. Den andra funktionen gör om datumet till antalet dagar, med AD tidstämpel, som kontot ska vara aktivt.

Användaren får se en sammanställning av de konton som kommer att aktiveras, med information om antal konton och för vilka gäster det kommer att aktiveras, se Figur 3.8.

Angivet syfte för kontona och de antal dagar de kommer vara aktiva visas. De angivna e-postadresserna kontrolleras så att den har rätt syntax, men då de inte är obligatoriska meddelas det bara om de har fel syntax. Meddelandet anger vilken e-postadress som är felaktig och på vilken rad den finns. Användaren får då välja att själv gå tillbaka och rätta till e-postadressen eller att gå vidare utan att någon information skickas. Det skrivs också ut

Figur 3.8: Sammanställning av konton.

hur många som inte har angiven e-postadress. I de fall ingen e-postadress angivits så skrivs en standard e-postadress in i det fältet. Denna adress är *null@kau.se*. All informationen från formuläret, tillsammans med lösenord, antal konton och antal dagar de ska vara aktiva skrivs in i ett dolt HTML-formulär för att skickas vidare när användaren är nöjd med sammanställningen. Med ett dolt HTML-formulär menas att användaren inte ser fälten utan de fylls i av systemet.

Efter att användare godkänner sammanställningen visas en sida med kontoinformation bestående av kontonamn, lösenord och för vilken gäst ett konto gäller för, se Figur 3.9.

En funktion anropas som returnerar ett slumpmässigt filnamn till en tom textfil och netID på inloggad användare hämtas. En iteration används för att från formuläret hämta information om varje enskilt konto. I denna iteration skickar en funktion kontonamn och lösenord till de e-postadresser som finns angivna. E-postmeddelandet innehåller också information om var kontot gäller samt instruktioner för hur det ska användas. Filnamnet, netID

Figur 3.9: De aktiverade kontona.

och informationen skickas som parametrar till en funktion, *activate_user*, som hämtar ett avaktiverat konto från AD. En funktion anropas som aktiverar det kontot som hämtades och sätter det aktuella lösenordet. Efter aktiveringen skriver funktionen *activate_user* in gästnamn, e-postadress, syfte, netID och avaktiveringsdatum som attribut i kontot. En annan funktion anropas som skriver in kontonamn, gästnamn, e-postadress, avaktiveringsdatum och lösenord i textfilen. Denna textfil används senare för att skapa de olika PDF-filerna. Ett sista funktionsanrop skriver in netID, kontonamn, avaktiveringsdatum, gästnamn, e-postadress och syfte i databasen. Funktionen *activate_user* returnerar sedan det kontonamn som hämtades för att det ska kunna skrivas ut. Det finns fyra funktioner som används för att skapa PDF-filer. En skapar en sammanställning av all kontoinformation i en tabell. De tre andra skapar PDF-filer med kontonamn, lösenord och gästnamn för varje enskilt konto som en egen sida. Dessa dokument har också instruktioner för användandet av kontot. Skillnaden mellan de tre funktionerna är att instruktionerna är på svenska, engelska respektive svenska och engelska. PDF-filerna skickas som e-postbilagor

till den användare som är inloggad. NetID på inloggad användare används som namn i e-postadressen. Länkar till dessa PDF-filer finns också så användaren kan spara eller öppna dem direkt.

Se/ändra konton En inloggad användare som har aktiverat konton kan hantera dessa genom att följa länken se/ändra konto. Här visas som standard alla konton som aktiverats av användaren och som fortfarande är aktiva, se Figur 3.10.

Figur 3.10: Se/ändra konton.

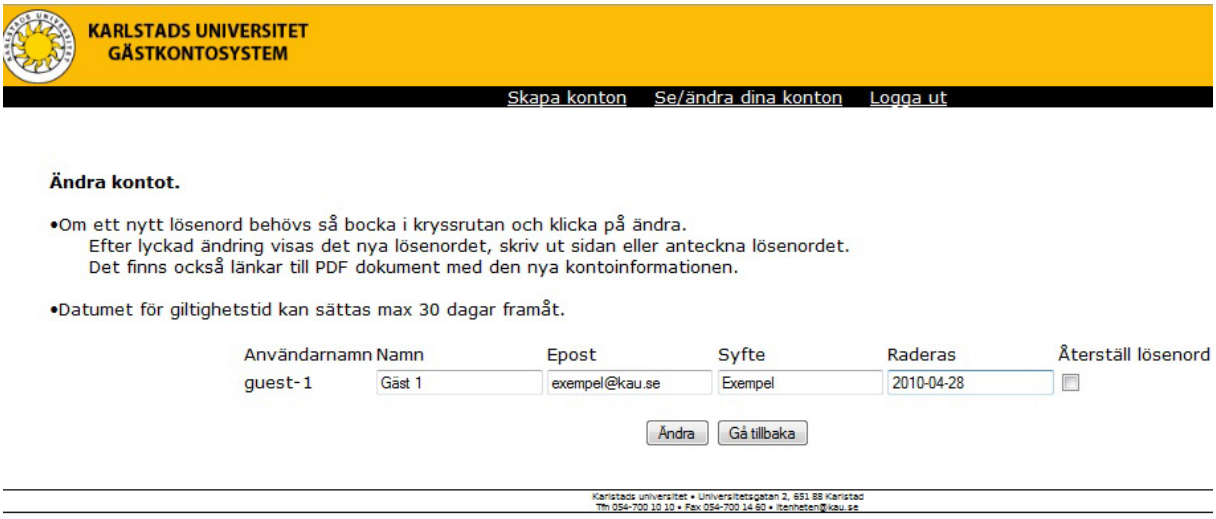
Kontoinformationen hämtas från AD med en funktion som tar inloggad användares netID som argument. Funktionen använder netID och att kontot ska vara aktivt som ett filter när den söker igenom AD, den skriver sedan ut resultatet i en tabell.

På sidan finns ett HTML-formulär där användaren kan ange olika sökkriterier för sina konton. I en rullista kan ett sökattribut väljas och i textfältet anges ett sökord. De val som finns är alla fält, gästnamn och syfte. Alla sökningar som sker har netID som en del av sökfiltret för att bara lista egna konton. Sökningarna sker också med jokertecken så att det går att söka med delsträngar. Om alla fält väljs, kommer sökordet användas som filter för attributen kontonamn, gästnamn, e-postadress och syfte. Om gästnamn eller syfte

väljs så är det dessa respektive attribut som används som filter. Alla sökfunktioner skriver ut resultatet i en tabell där varje rad innehåller två HTML-formulär. Båda formulären innehåller var sin knapp och ett dolt fält som plockar ut kontonamnet på aktuellt konto.

Det ena formuläret används för att avaktivera kontot. En funktion hämtar gästnamn och syfte från kontot på AD och anropar en funktion som avaktiverar det. Avaktiveringen sker genom att ändra attributet *UserAccountControl* i kontot. Om det lyckas, så skrivs kontonamnet, gästnamnet, netID, syfte, dagens datum och att kontot blir inaktiverat som en post i databasen och ett meddelade visas för användaren att kontot är avaktiverat. Om något blev fel och kontot inte avaktiveras meddelas detta.

Det andra formuläret tar användaren till en sida där attributen på ett kontot kan ändras. Attributen som kan ändras är gästnamn, e-postadress, syfte, avaktiveringsdatum och lösenord. Sidan där ändringarna kan göras består av ett HTML-formulär som innehåller fyra textfält, en kryssruta och två knappar, se Figur 3.11. De attribut som finns på kontot



Figur 3.11: Ändra konton.

hämtas och skrivs in i respektive textfält och användaren kan då göra önskade ändringar. Om lösenordet ska ändras markeras kryssrutan för detta. När användaren gjort sina ändringar och går vidare kontrolleras det som står i textfälten. I gästnamn, e-postadress och

syfte tas de flesta specialtecknen bort. Om kryssrutan för lösenord är markerad genereras ett nytt lösenord. Fälten namn och syfte kontrolleras, så att de inte är tomma. Datum kontrolleras så att det har rätt format, inte har passerats och inte ligger mer än 30 dagar framåt i tiden. Om något blir fel i dessa kontroller, skickas användaren tillbaka till se/ändra konto sidan och ett felmeddelande skrivs ut, till exempel att fältet syfte är tomt. När ändringarna är godkända skrivs de till AD. Det skrivs också en post till databasen med information om ändringen. När detta sker så skrivs kontonamn, gästnamn, eventuellt lösenord och avaktiveringsdatum till en textfil som sedan används för att skapa PDF-filer. Till sist kontrolleras syntaxen på e-postadressen. Efter detta kommer användaren till en sida som meddelar om ändringen lyckades eller inte, se Figur 3.12. Om det finns syntaxfel

Figur 3.12: Lyckad ändring med nytt lösenord.

på e-postadressen så meddelas det här, men användaren får själv avgöra om den ska ändras på nytt då den inte är obligatorisk. I detta läge skickas ingen information till gästen via e-post. Kontonamnet på aktuellt konto visas och om ett nytt lösenord är satt visas även det. Om lösenordet inte är bytt så står det att det tidigare lösenordet gäller istället. PDF-filer med den nya kontoinformationen skapas och visas som länkar så att användaren kan öppna eller spara härifrån. PDF-filerna skickas inte via e-post till användaren i detta

läget.

Längst ner på sidan se/ändra konton finns två knappar som påverkar alla konton i tabellen. En för att ändra alla lösenord och en för att avaktivera alla konton. När någon av dessa används visas en pop-up-fönster där användaren får bekräfta åtgärden. Funktionen för att avaktivera alla konton liknar den som avaktiverar enskilda konton. Skillnaden är att antalet konton i tabellen räknas och avaktiveringen sker i en iteration som avaktiverar kontona ett och ett. Ett meddelande skrivs ut som talar om ifall kontona blev avaktiverade eller inte och alla konton som fortfarande är aktiva skrivs ut. Funktionen som ändrar alla lösenord går också igenom de aktuella kontona ett och ett i en iteration. I iterationen genereras ett nytt lösenord och detta skrivs till kontot. De attribut som behövs till databasen och PDF-filer hämtas från AD. Den uthämtade datan skrivs till databasen med information om att lösenordet har ändrats. Det nya lösenordet, kontonamn, gästnamn och avaktiveringsdatum skrivs till den temporära textfilen som är till för kunna skapa PDF-filerna. När ändringarna är gjorda skrivs kontonamnen och de nya lösenorden ut i en tabell och PDF-filer skapas som ovan. Om något gick fel för något konto skrivs detta ut istället för lösenordet och i PDF-filerna är lösenordet blankt.

3.3.3 Administratörsdel

När en administratör har loggat in via CAS visas en informationssida, se Figur 3.13.

Här står hur många fördefinierade konton som finns i AD och hur många av dessa som för tillfället är aktiverade. Denna information hämtas genom att koppla upp mot ett administratörskonto i AD och sedan använda en funktion som söker igenom AD med olika filter. Den här kopplingen används vid all kommunikation med AD i följande funktioner. Det som kan göras är:

- Lista aktiva konton och avaktivera konton.
- Visa logginformation om en viss användare.

Figur 3.13: Startsida för en administratör.

- Visa logginformation om ett visst konto.
- Logga ut från applikationen.

Länkar på sidan leder till de olika funktionerna. Dessa länkar finns på alla sidor som administratören kan komma till. Funktionerna inklusive utloggningen finns därmed tillgängliga oavsett vad som gjorts tidigare.

I ”Lista/Avaktivera konton” skrivs som standard alla aktiva konton ut i en tabell, se Figur 3.14. Information skrivs ut med en funktion som söker igenom AD och för varje aktivt konto hämtar och skriver ut attributen: kontonamn, netID på den som aktiverat kontot, syftet och avaktiveringsdatum.

Det finns en sökfunktion på sidan som söker i AD på attributen: kontonamn, netID och syfte. Med denna funktion kan man till exempel lista alla konton som har ett gemensamt syfte. Informationen som visas är den samma som i Figur 3.14. På varje rad i tabellen skapas ett HTML-formulär med en knapp för att avaktivera kontot på den raden.

När administratören avaktiverar ett konto, så hämtas kontonamnet från formuläret. Utifrån detta kontonamn hämtas attributen gästnamn och syfte från AD, sedan avaktiveras kontot. Avaktiveringen sker genom att ändra attributet *UserAccountControl* i kontot. En

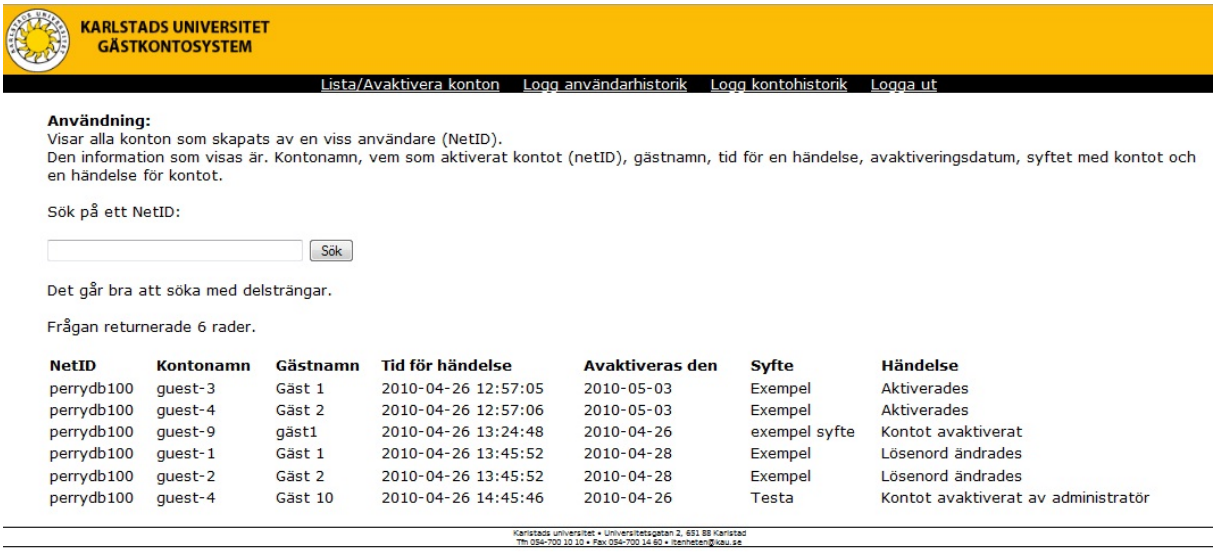
Figur 3.14: Lista aktiva konton och avaktivera konton.

post skrivs till databasen som innehåller kontonamn, administratörens netID, aktuellt datum, gästnamn, syfte och att kontot avaktiverats av en administratör.

Sist på sidan finns en knapp som avaktiverar alla konton i tabellen. Denna avaktiveringsfunktion fungerar på samma sätt som ovan, med den skillnaden att flera konton avaktiveras genom en iteration. När en knapp för avaktivering används så visas ett pop-up-fönster där administratören får bekräfta avaktiveringen. Detta gäller både för avaktivering av enskilda konton och för en lista med konton.

I de båda loggfunktionerna kan en administratör få information om exempelvis vad som hänt med ett visst konto eller vem som aktiverade kontona och när detta hände. Denna information hämtas från databasen som används som logg för systemet. Loggning sker för att få spårbarhet i hur systemet används, vilket var ett av kraven i specifikationen. I logginformation om användare finns en sökfunktion där administratören anger ett netID i ett textfält, se Figur 3.15.

Sökningen kan ske med delsträngar om det fullständiga netID inte är känt. Från databasen hämtas alla poster där fältet netID matchar sökningen. Informationen som visas är netID, kontonamn, gästnamn, tid för en händelse för ett konto, avaktiveringsdatum, syfte



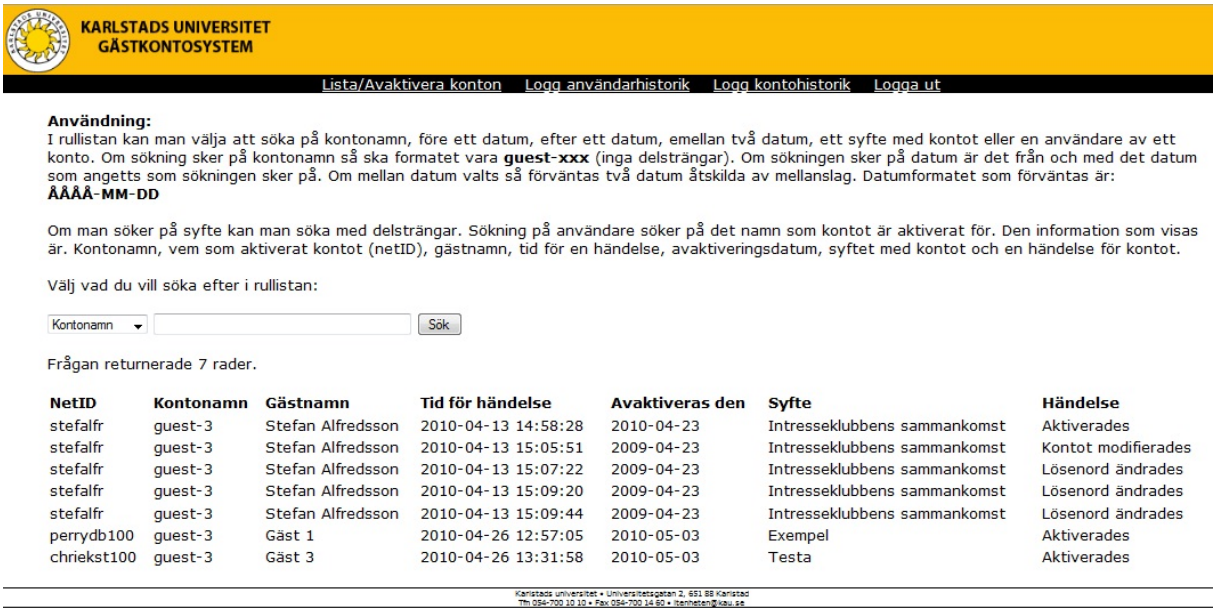
Figur 3.15: Logginformation om användare.

och en händelse för ett konto. Detta skrivs ut som en tabell.

I logginformation om konton finns en sökfunktion där administratören kan välja sökattribut från en rullista och ange en söksträng i ett textfält, se Figur 3.16. De sökattribut som finns är kontonamn, före ett datum, efter ett datum, mellan två datum, gästnamn och syfte med kontot. Vid sökning på kontonamn är det hela kontonamnet som gäller. Detta för att undvika att lista alla konton på till exempel 100-tal om sökningen sker på guest-1. På de övriga sökattributen kan delsträngar användas. Beroende på vilket attribut som väljs så matchas söksträngen mot motsvarande fält i databasen. Informationen visas på samma sätt som i Figur 3.16.

3.4 Kapitelsammanfattning

I detta kapitel har vi beskrivit analysen och designen av vårt examensarbete. Analysen bestod av att ta fram en kravspecifikation för systemet och sedan diskutera hur den kan implementeras. Vi har beskrivit de delar som ingår i designen av systemet. Dessa är AD, CAS, en användardel, en administratörsdel och en databas. Vi har beskrivit hur vår databas



Figur 3.16: Logginformation om konto.

ser ut och strukturen på AD. Vi har gått igenom vilka funktioner som finns för användare och administratörer. För en användare är dessa att aktivera gästkonton i AD och att hantera de aktiverade kontona. En administratör kan avaktivera konton och se en logg över användandet av systemet. Det som finns i loggen är vad en specifik användare har gjort och vad som hänt med ett givet konto. Vi har beskrivet hur systemet är tänkt att användas.

4 System- och användartest

Vid utveckling av nya program och system är testning en central del [31]. Det är viktigt att det nya systemet blir ordentligt testat innan det leveras till kund. Detta för att säkerställa att det lever upp till kundens förväntningar och krav. Det lönar sig att börjat testa tidigt i utvecklingsprocessen. Vi har under examensarbetes gång lagt mycket tid på testning, för att säkerställa att systemet ska vara så felfritt som möjligt. Det går dock aldrig att få ett

system helt felfritt och helt säkert mot angripare. Det går dock att med systematiska tester minska risken för att det ska finnas felaktigheter i systemet. I nästa delkaptiel kommer vi presentera den testning som har skett av systemet dels under utvecklingstiden och dels efter.

4.1 Testning

Vi har under tiden vi utvecklade systemet kontinuerligt testat det. I början av utvecklingen testade vi funktion för funktion som utvecklades så kallad enhetstestning [1], för att säkerställa att varje enskild funktion fungerade som tänkt. Vi har även gjort mycket utforskade testning [5] som även kallas ad hoc testning [3] av systemet. Testning ledde fram till att ett flertal fel upptäcktes och kunde åtgärdas. Systemet blev sedan testat av personal på Karlstads universitet. Det resulterade i att ytterligare några fel upptäcktes. Alla de fel som har hittas under testningen har åtgärdats.

Det finns dock en brist kvar i systemet, som inte är ett fel i sig utan en säkerhetsrisk. När en användare har aktiverat eller ändrat konton så genereras det PDF-filer med kontoinformationen. Dessa PDF-filer finns kvar på webbservern när användaren är klar och har loggat ut. Det innebär att PDF-filerna är tillgängliga för vem som helst bara filnamnet är känt. Vilket innebär att vem som helst kan komma åt uppgifter om konton som en annan användare aktiverat. Vi har därför försökt göra filnamnen svårgissade. Filnamnet är uppbyggt av fyra olika textsträngar. Först i namnet är det ursprungliga filnamnet för att veta vilken variant av PDF-fil det är. Efter filnamnet så kommer en md5hash [30] av Unix-tiden [34] när filen skapas. NetID på användaren som aktiverade kontot och dagens datum plus en dag. Ett exempel är `users_all_3ac9c36042d3324289b235dc065b0c29_chriekst100_2010-05-05.pdf`

- `users_all` är det ursprungliga filnamnet.
- `3ac9c36042d3324289b235dc065b0c29` är md5hashen.

- `chriekst100` är det NetID som skapade filen.

- `2010-05-05` datumet när filen skapades plus en dag.

Datumet är tänkt att används för att via en schemalagt skript på webbservern kunna radera gamla PDF-filer.

Vidare finns det en funktion i systemet som inte har gått att testa. Det är anropet till tjänsten som kontrollerar om ett netID tillhör en student eller anställd på universitetet. Vi har inte lyckats komma åt den tjänsten från testmiljön. Trots ihärdiga försök av flera olika personer på IT-avdelningen så kommer vår testserver inte åt den berörda tjänsten. Koden har istället granskats av personal på IT-avdelningen och de anser att funktionen kommer att fungera i en skarp miljö. Eventuella problem får åtgärdas vid en skarp implementation.

Administratörsdelen har inte blivit så genomtestad som användardelen. De funktioner som administratören har tillgång till har testats så att de funktionsmässigt fungerar. Det har dock inte i någon större utsträckning skett någon aktiv testning för att hitta fel. Detta beror på att den personal som har testat systemet bara har haft tillgång till användardelen. I administratörsdelen av systemet vet vi att en SQL-injektion [18] kan göras, men då en administratör ändå ser allt i databasen ansåg vi att detta inte spelar någon roll.

4.2 Kapitelsammanfattning

I det här kapitlet har testning diskuterat. Vi har beskrivit testning i allmänhet och de testmetoder vi har använt vid testningen av systemet. Vi har presenterat en säkerhetsbrist med PDF-filerna som upptäcktes vid testningen. Den lösningen vi valde har presenterats och förklarats. Vi har även tagit upp den funktion som tyvärr inte har blivit testad. På grund av att testmiljön vi har använt inte har åtkomst till den servern i den skarpa IT-miljön som behövs för att testa funktionen som avgör om det är student eller anställd som loggar in.

5 Diskussion

I detta kapitel tar vi upp de problem som uppstått under utvecklingen av systemet. Vi diskuterar alternativa lösningar på vissa detaljer i implementationen. Vi tar upp alternativ till hur kontoinformation ska anges av användaren och hur den skrivs till loggen. Hur lösenord kan genereras och vilket konto som väljs för aktivering i AD. För- och nackdelar med att mellanlagra kontoinformation i en databas och hur konton ska aktiveras. Till sist tar vi upp förslag på framtida utveckling.

5.1 Problem

I detta delkapitel tar vi upp problem som uppstått under utvecklingen av systemet. Det gäller både problem med testmiljön och implementationsmässiga problem. Vi presenterar även lösningarna på problemen i de fall där problemet har blivt löst.

5.1.1 Installera AD

När vi skulle sätta upp testmiljön så fick vi problem när vi skulle konfigurera AD och DNS-servern på en av de virtuella servrarna. Vi fick inte den virtuella klienten att kommunicera med AD, vilket medförde att det inte gick att ansluta klienten till domänen. Vi löste problemet genom att konfigurera om AD och DNS-servern med hjälp av boken Windows Server 2008 Active Directory Resource Kit [29], samt specificera att DNS servern ska användas av den virtuella klienten.

5.1.2 Certifikat/LDAP

Vi hade stora problem att få de två virtuella servrarna att kommunicera. För att kunna utföra flertalet operationer över LDAP, så kräver AD en säker uppkoppling med TLS [13]. För att kunna koppla upp en säker anslutning krävs certifikat [11]. Ett certifikat kan beskrivas som en digital legitimation som används för att kunna kontrollera om det går

att lita på en server som kommunikation sker med. Då verifierade certifikat kostar pengar var det aldrig ett alternativ. Vi lyckades efter mycket letade skapa egensignerade certifikat, men de ville inte AD-servern acceptera. Det löste sig tillslut genom att vi på webbservern skapade en konfigurationsfil som gör att servrarna inte bryr sig om att certifikaten är egensignerade eller ogiltiga. Detta är ett problem som bara har betydelse i testmiljön vi har använt. I den skarpa miljön är det inget problem då de skarpa servrarna har verifierade certifikat.

5.1.3 Tjänsten för att kontrollera NetID

Vi har inte lyckats få vår virtuella webbserver att kommunicera med en av IT-avdelningens servrar. Den kommunikationen är nödvändig för att kunna kontrollera om ett netID tillhör en student eller personal på universitetet. Trots ihärdig felsökning av flera ur IT-avdelningens personalstyrka, så löstes aldrig problemet. Misstanken är att det är en router någonstans på vägen i nätverket som stoppar de specifika paketen. Personalen anser inte att detta ska vara ett problem i den skarpa miljön.

5.1.4 Testservern

Vi har haft problem med att testservern har fått flertalet Blue Screen of Death (BSoD) [33]. Enligt felmeddelandet så är anledningen antingen en felaktig drivrutin eller fel på hårdvaran. Vi misstänker att den har att göra med att en av handledarna på IT-avdelningen bytte ut grafikkortet i testservern utan att ta bort de gamla drivrutinerna. Vid installation av nya drivrutiner till det nya grafikkortet har det uppstått en konflikt som har orsakat flertalet BSoD.

5.1.5 Implementationsproblem

Det har varit många problem under implementationen. Några som kan vara värda att nämna är:

PDF-filerna Det var problem med att få svenska tecken att fungera i PDF-filerna. Det löstes genom att byta teckenkodning på alla sidor i systemet som används vid skapandet av PDF-filerna. Teckenkodningen byttes till `iso-8859-1`.

Kontrollera indata Den inbyggda PHP-funktionen `preg_match` används för att kontrollera indata. Problemet var att vi inte fick den att kontrollera svenska tecken, vilket resulterade i att det inte gick att ange enbart svenska tecken som ett namn eller syfte. Det är i sig inget stort praktiskt problem, men väldigt irriterande att det inte fungerar. Att funktionen inte returnerade något felmeddelande gjorde det inte lättare att lösa problemet. Vi löste det dock genom att byta teckenkodning till `utf-8` på variablerna innan de kontrollerades av `preg_match`.

Lösenordsfunktionen Vi har utvecklat en egen funktion som genererar slumpmässiga lösenord. Detta på grund av att det inte finns någon inbyggd funktion i PHP för att generera lösenord med den komplexitet som behövs. Problemet med funktionen är att den ibland genererar samma lösenord flera gånger på rad. Även antalet gånger den genererade strängen upprepar sig varierar. Vi har inte löst det här problemet utan låter det vara kvar. Det bör dock lösas innan systemet tas i skarp drift.

Lösenordsbyte vid aktivering av gästkonto Det var mycket svårt att få AD att acceptera nya lösenord till gästkontona. AD är mycket känslig för i vilket format lösenorden ska vara i. Vi försökte länge skriva en funktion för att kunna byta lösenord på konton utan att lyckas. Det slutade med att vi via Google [23] hittade en fungerade funktion för att byta lösenord.

Hantering av tid i AD AD lagrar tid som antalet 100 nanosekundintervall som har förflutit sedan den 1 januari 1601. Det innebär för att kunna sätta ett sista giltighetsdatum på ett gästkonto så måste det datumet räknas om till AD-tid. Det finns ingen inbyggd

omvandlare i varken PHP eller AD. Det gjorde att vi var tvungna att skriva en egen algoritm som gör den omvandlingen.

5.2 Alternativa lösningar

En del förslag till ändringar av systemet har framkommit under utvecklingen. En del av dessa har vi själva kommit fram till och en del har den personal som hjälpt till med testningen kommit med.

Datumet som anges för hur länge ett konto ska vara aktiverat kan med fördel bytas ut mot en rullista där användaren kan välja antal dagar från 1 till 30. Detta gör det enklare för en användare att ange antalet dagar som ett konto ska vara aktivt. Ett annat alternativ vore att ha en kalenderfunktion på sidan där användaren kan markera ett avaktiveringsdatum.

Lösenorden som nu genereras slumpmässigt kan genereras så att de blir uttalbara för att öka användarvänligheten för gästen.

Istället för att skriva direkt till AD kan informationen mellanlagras i en databas för att sedan skrivas till AD. En skillnad blir att konton som mellanlagras aktiveras på en specifik tid istället för att, som konton i vårt system, aktiveras direkt. Detta medför att de mellanlagrade kontona inte finns tillgängliga för gästen när användaren aktiverar dem. En fördelen med mellanlagring är att en användare kan ange ett aktiveringsdatum och på så sätt ha framförhållning på sina gästkonton. Det finns också några nackdelar med detta. En är att lösenorden till konton måste lagras i klartext eller mellankrypteras i databasen för att finnas tillhands vid aktiveringen. En annan är att användaren inte får någon bekräftelse om aktiveringen i AD lyckades eller inte. Det går heller inte att visa kontoinformationen direkt för användaren.

I vårt system får en användare vänta då aktiveringen sker i AD för att sedan få en bekräftelse om aktiveringen lyckades eller inte. Väntetiden är maximalt 30 sekunder då detta är standard för längsta exekveringstid i PHP. Kontonamn och lösenord för de aktiverade kontona visas efter aktiveringen. Ett alternativ till detta vore att låta aktiveringen

ske schemalagt och på så sätt få bort väntetiden. Det innebär att kontoinformationen måste mellanlagras och allt det medför, se ovan.

Ett alternativ till att ange gästinformation är att användaren först anger ett antal konton som ska aktiveras. Sedan skapas två textboxar för varje enskilt konto, en för gästnamnet och en för e-postadressen. En fördel med detta är att det blir lättare implementationsmässigt att skilja på namn och e-postadress då dessa finns i olika fält. Detta implementerades inte då det i analysen ansågs bättre med en textarea så användaren kan klistra in information om fler gäster åt gången, till exempel från ett Excel-dokument.

Vårt system plockar ut det inaktiva kontot med lägst nummer först då en användare vill aktivera konton. Detta kan leda till segmentering av kontonamn. Det vill säga att en användare som aktiverar fler konton inte får på varandra följande kontonamn, till exempel guest-001, guest-007 och så vidare. Det medför också att om någon vill knäcka ett lösenord, är det mest troligt att ett konto med lågt nummer är aktivt. Ett alternativ till detta vore att ta de kontonamnen som finns i följd och som räcker till de antal konton som ska aktiveras vid ett tillfälle. Detta gör att segmentering av kontonamn blir så liten som möjligt. Ett annat alternativ vore att plocka kontonamn slumpmässigt så det blir svårare att veta vilka konton som är aktiva.

5.3 Framtida utveckling

Det har under arbetets gång utvecklats förslag och idéer på framtida utveckling av applikationen. Vi presenterar här de förslag och idéer som vi och personer på IT-avdelningen har kommit på.

Vi fick önskemål från IT-avdelningen att titta på en lösning för att skicka kontoinformation till gäster via SMS. Här har vi hittat några möjliga alternativ. Ett är att prenumerera på en tjänst som tar emot ett vanligt e-postmeddelande som har ett mobilnummer som namn. Detta vidarebefodras sedan som SMS till angivet nummer via en SMS-gateway. Det finns andra liknande tjänster som tar emot meddelande av olika typ och vidarebefordrar

dem. Det går också att sätta upp en egen SMS-gateway. För detta alternativ finns lösningar baserade på öppen källkod eller företag som säljer färdiga system.

En utveckling av systemet vore att ge en användare möjlighet att ändra det angivna syftet för samtliga konton med gemensamt syfte. Det samma gäller för avaktiveringsdatum för kontona. I nuvarande system kan användaren bara ändra dessa attribut för ett konto i taget.

Där kontoinformationen anges av användaren kan det läggas till ett formulär som ger användaren möjlighet att importera gästinformationen från till exempel en textfil. Det innebär att användaren inte behöver kopiera över informationen till HTML-formuläret i applikationen utan kan ladda upp en fil med informationen.

När en e-postadress till en gästanvändare ändras tillsammans med att ett nytt lösenord genereras så kan den nya informationen skickas till den nya e-postadressen. Detta kan göras så det sker automatiskt eller att användaren till exempel får markera en kryssruta. Det här kan även läggas till som funktion då samtliga lösenord ändras.

De olika loggarna som administratören kan se skulle också kunna visas i grafisk form för att göra den mer överskådlig, till exempel i diagram. Nu presenteras all information från databasen i tabeller.

Funktioner som möjliggör att administratören kan ta bort gamla poster i databasen kan läggas till. Detta för att det med tiden kan bli mycket information att överblicka och att informationen efter en viss tid är mindre relevant. En alternativ lösning för detta vore med ett skript på servern som tar bort posterna, till exempel då de legat i ett antal månader.

I vårt system visas och används netID för användaren som namn i e-postadressen. Detta kan bytas ut mot det alias som finns för användaren, vilket gör det lättare för en användare att känna igen sin egen e-postadress.

Ett tillägg är att ta hand om datan från inmatningsfälten på administratörsdelen. Vi kontrollerar inget av den datan idag, så en SQL-injektion är möjligt som administratör. Detta kan ses som en säkerhetsbrist. Att ingen kontroll görs beror på att en administratör

ändå har tillgång till all information i databasen.

En säkerhetsbrist som finns är att de PDF-filer som skapas finns kvar på webbservern. Det betyder att en användare kan komma åt PDF-filer som tillhör en annan användare och därigenom också få tillgång till kontoinformation. För att göra detta måste filnamnet vara känt. En lösning på detta vore att ha ett skript på webbservern som tar bort PDF-filerna vid en bestämd tidpunkt. Ett datum har därför lagts in i filnamnen för att veta när en fil kan raderas. Även om ett raderingsskript används så kvarstår problemet under en viss tid.

5.4 Kapitelsammanfattning

I detta kapitel har vi tagit upp de problem som vi har haft under utvecklingen av applikationen. En del av de problem som tagits upp har med testmiljön att göra i form av att sätta upp den och BSoD. Andra problem som tagits upp är de som har med implementationen av systemet att göra. Vi har också presenterat ett antal alternativa lösningar till de vi har implementerat. Vi har också gett en del förslag på framtida utveckling av systemet. Vissa av dessa anser vi är förbättringar av systemet och andra enbart alternativa lösningar. Detta är för att utöka funktionaliteten för både användare och administratörer. Några av dessa uppdateringar har med säkerheten i systemet att göra, dock inga som vi anser som allvarliga.

6 Slutsats

Målet med vårt examensarbete var att skapa en webbapplikation för att hantera ett gästkontosystem på Karlstads universitet. En del av arbetet var dessutom att ta fram en detaljerad kravspecifikation med vilka funktioner som systemet skulle innehålla. Systemet som vi skapade består av ett AD, en webbserver, en databas och universitetets inloggningssystemet CAS. På AD finns det ett OU med fördefinierade konton som används som gästkonton. På webbservern finns vår applikation som manipulerar gästkonton. Kontakten

mellan webbserver och AD sker över LDAP. För att komma åt applikationen måste inloggning ske via CAS. De funktioner som finns är att en användare kan aktivera konton. När detta sker får användaren information om de kontonamn, med tillhörande lösenord, som aktiverats. Denna information skickas också som e-postbilagor i PDF-format. De modifikationer som görs i AD skrivs också till en databas som används som logg för att få spårbarhet i systemet. Denna logg finns tillgänglig för en administratör av systemet.

Under arbetets gång har vi lärt oss mycket om att arbeta med katalogtjänster. Detta tycker vi är positivt då katalogtjänster används i många av dagens IT-system. Vi har även fått bättre kunskap om PHP då detta språk i huvudsak har använts för implementationen av systemet. Övriga språk som använts är HTML, CSS och JavaScript.

Det har också framkommit förslag på förbättringar och alternativa lösningar under arbetet. De förbättringar som föreslagits är mestadels utökning av funktionaliteten av systemet. En föreslagen funktion var att kontoinformationen skickas till gästen via SMS. En annan är att använda ett alias istället för netID för en användare. De förslag som vi inte ansett vara förbättringar finns beskrivet som alternativa lösningar. De har till exempel att göra med hur information anges av en användare och vid vilken tid ett konto ska aktiveras.

Vi anser att vi uppnått målet med examensarbetet då vi skapat en väl fungerande prototyp av systemet. Vårt system uppfyller kraven från den preliminära specifikationen och den framtagna detaljerade kravspecifikationen. Systemet är testat både av oss och av personal på Karlstads universitet och de fel som framkommit i dessa tester är åtgärdade.

Referenser

[1] IEEE standard for software unit testing. Technical report, 1986. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=27763, besökt 2010-05-17.

[2] Mehdi Achour, Friedhelm Betz, Antony Dovgal, Nuno Lopes, Hannes Magnusson, Georg Richter, Damien Seguy, and Jakub Vrana. PHP Manual. Website. http://www.php.net/manual/en/, besökt 2010-05-17.

[3] Chris Agruss and Bob Johnson. Ad hoc Software Testing, 2000. http://www.testingcraft.com/ad_hoc_testing.pdf, besökt 2010-05-17.

[4] The Apache Software Foundation. Apache HTTP Server. Website. http://www.apache.org, besökt 2010-05-17.

[5] J. Bach. Exploratory Testing Explained. 2002. http://www.cs.uni.edu/~wallingf/teaching/172/resources/exploratory-testing.pdf, besökt 2010-05-17.

[6] Bert Bos, Tantek Celik, Ian Hickson, and Håkan Wium Lie. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, 2009. http://www.w3.org/TR/CSS21, besökt 2010-05-17.

[7] Romain Bourdon. WampServer. Website. http://www.wampserver.com/en, besökt 2010-05-17.

[8] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), October 1989. Updated by RFCs 1349, 4379.

[9] D. W. Chadwick. *Understanding X.500 (The Directory).* International Thompson Publishing, 1996. http://sec.cs.kent.ac.uk/x500book/, besökt 2010-05-17.

[10] The World Wide Web Consortium. Website. http://www.w3.org/, besökt 2010-05-17.

[11] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.

[12] Microsoft Corporation. Website. http://www.microsoft.com, besökt 2010-05-17.

[13] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008.

[14] Mary Feeney. *The Standard Generalized Markup Language (SGML).* British Library Research and Development Dept. and Library & Information Technology Centre, London, UK, 1988.

[15] Cisco Secure Access Control Server for Windows. Website. http://www.cisco.com/en/US/products/sw/secursw/ps2086/, besökt 2010-05-17.

[16] The Internet Engineering Task Force. Website. http://www.ietf.org, besökt 2010-05-17.

[17] James Groff and Paul Weinberg. *SQL The Complete Reference, 3rd Edition*. McGraw-Hill, Inc., New York, NY, USA, 2010.

[18] W. Halfond and A. Orso. *Malware Detection*, volume 27 of *Advances in Information Security*, chapter Detection and Prevention of SQL Injection Attacks. Springer, 2007.

[19] Stefan Hinz, Paul DuBois, Jonathan Stephens, Martin 'MC' Brown, and Anthony Bedford. MySQL Documentation: MySQL Referens Manuals, 2010. http://dev.mysql.com/doc, besökt 2010-05-17.

[20] B. Hoehrmann. Scripting Media Types. RFC 4329 (Informational), April 2006.

[21] Timothy A. Howes, Mark C. Smith, and Gordon S. Good. *Understanding and Deploying LDAP Directory Services*. New Riders Publishing, Cambridge, Massachusetts, 1998.

[22] Jani Hursti. Single sign-on. 1997. http://www.tml.tkk.fi/Opinnot/Tik-110.501/1997/single_sign-on.html, besökt 2010-05-17.

[23] Google Inc. Website. http://www.google.com/about.html, besökt 2010-05-17.

[24] Jasig. Website. http://www.jasig.org, besökt 2010-05-17.

[25] Sun Microsystems. VirtualBox. Website. http://www.virtualbox.org, besökt 2010-05-17.

[26] Adobe och PDF. Website. http://www.adobe.com/se/products/acrobat/adobepdf.html, besökt 2010-05-17.

[27] Andrew Petro and Mark Rogers. Central Authentication Service (CAS). Website. http://www.ja-sig.org/wiki/display/CAS/Home, besökt 2010-05-17.

[28] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. HTML 4.01 Specification. Website, 1999. http://www.w3.org/TR/html401, besökt 2010-05-17.

[29] Stan Reimer, Conan Kezema, and Mike Mulacare with the Microsoft Active Directory Team. *Windows Server 2008 Active Directory Resource Kit*. Microsoft Press, Redmond, Washington, USA, 2008.

[30] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321 (Informational), April 1992.

[31] T. Ryber. *Testdesign för programvara.* Miguru Media AB, 2006.

[32] Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating System Concepts.* Wiley, December 2004.

[33] Demystifying the 'Blue Screen of Death'. Website. `http://technet.microsoft.com/en-us/library/cc750081.aspx`, besökt 2010-05-17.

[34] Unix time. Website. `http://en.wikipedia.org/wiki/Unix_time`, besökt 2010-05-17.

[35] Organizational units. Website. `http://technet.microsoft.com/en-us/library/cc758565.aspx`, besökt 2010-05-17.

[36] Active Directory Users and Computers. Website. `http://technet.microsoft.com/en-us/library/cc754217.aspx`, besökt 2010-05-17.

A Systembeskrivning

Den här bilagan innehåller en översikt över samtliga filer och funktioner i systemet.

checkfunctions.php Innehåller fem hjälpfunktioner till systemet.

- *adDag($datum)*
  - $datum - Datumet som användaren har angett som sista giltighetsdatum för ett gästkonto.

  Räknar ut antalet dagar mellan dagens datum och det datum som en användare angett och returnerar antalet.

- *clearTempTextFile()* Skapar en tom textfil med ett slumpgenererat namn och returnerar därefter filnamnet.

- *clearTextFile($name)*
  - $name - Namnet på filen som ska tömmas på innehåll.

  Tömmer en textfil på innehåll. Tar emot ett filnamn som parameter. Öppnar filen och raderar innehållet, sparar och stänger filen.

- *genPass()*
  Genererar ett tio tecken långt lösenord som returneras.

- *sendPdf($epost, $pdf1, $pdf2, $pdf3, $email_subject)*
  - $epost - Mottagarens e-postadress.
  - $pdf1 - filnamn på bilaga nummer ett.
  - $pdf2 - filnamn på bilaga nummer två.
  - $pdf3 - filnamn på bilaga nummer tre.

  - $email_subject - E-post-meddelandets ämne.

  Skickar ett e-post-meddelande med tre PDF-filer som bilagor. PDF-filerna innehåller kontoinformation om de konton som har skapats eller modifierats. Meddelandet skickas till den användaren som har gjort ändringarna.

- *skickaEpost($epost, $anv, $pass)*

  - $epost - Mottagarens e-postadress.
  - $anv - användarnamnet på gästkontot.
  - $pass - lösenordet på gästkontot.

  Skickar ett e-post-meddelande till en gästanvändare med inloggningsinformation till ett konto.

edituser.php Används för att ändra attribut på ett gästkonto. De attribut som kan ändras är: gästnamn, gästens e-postadress, syftet med kontot samt sista giltighetsdag. Det finns även möjlighet att återställda lösenordet på ett konto. Anropas från `changeAccount.php`. Tar emot en variabel innehållande ett CommonName via ett HTML-formulär från `changeAccount.php`. Skriver ut ett HTML-formulär med ifyllt med data som hämtas från AD med hjälp av det CommonName som kom i från `changeAccount.php`. När användaren trycker på ändra så kontrolleras datan i HTML-formuläret och är den korrekt så skrivs den till AD.

fpdf.php Finns att hämta på `www.fpdf.org` och innehåller klassen som används för att skapa PDF-filer.

getpdf.php Innehåller ett flertal villkorssatser för att ladda ner PDF-filer. Gör så att när en användare trycker på en länk till en PDF-fil så öppnas den inte utan användaren får välja på att spara ner den lokalt eller öppna den lokalt efter nerladdning.

index.php Startsidan i hela systemet. Inloggad användarens netID kontrolleras och sedan visas en meny beroende på om användaren är vanlig personal eller administratör.

kau_api_service_call.php Skriven av IT-avdelningen vid Karlstads Universitet. Innehåller *kau_api_service_call()* som kontrollerar om ett netID tillhör en anställd på universitetet eller inte.

- *kau_api_service_call($service_tag, $method, $params)*
  - $service_tag - Vilken tjänst som ska anropas.
  - $method - Vilken funktion som ska anropas.
  - $params - En associativ array.

  Kontrollerar om ett givet netID tillhör en anställd på universitetet eller inte.

loggaccount.php En administratörssida som ger administratören möjlighet att söka i loggen. Det går att söka på flera olika saker. Gästkontonamn, vad som har hänt före ett datum, mellan två datum, efter ett datum, ett specifikt syfte eller på ett specifikt namn (namn på en gäst).

logguser.php En administratörssida som ger administratören möjlighet att söka i loggen efter ett specifikt NetID. För att kunna kontrollera vad en given användare har använt för olika gästkonton.

mysql.php Innehåller funktioner för att skriva och läsa från MySQL-databasen.

- *addChangeToLog($netid, $username, $inaktiveras, $namn, $syfte, $info, $mail)*
  - $netid - Användaren som ändrade kontot.
  - $username - Namnet på gästkontot i AD.

- $inaktiveras - Datum när gästkontot ska inaktiveras.
- $namn - Gästens namn.
- $syfte - Syftet med kontot.
- $info - Vad som hände med kontot. Ändrades, lösenordet återställdes, avaktiveras av användare, avaktiveras av admin.
- $mail - Gästens e-postadress.

Skriver en post till databasen när ett konto har ändrats.

- *addToLog($netid, $username, $inaktiveras, $namn, $epost, $syfte)*
  - $netid - Användaren som aktiverade kontot.
  - $username - Namnet på gästkontot i AD.
  - $inaktiveras - Datum när gästkontot ska inaktiveras.
  - $namn - Gästens namn.
  - $epost - Gästens e-postadress.
  - $syfte - Syftet med kontot.

  Skriver en post till MySQL-databasen när ett konto har aktiverats.

- *logglist($num, $result)*
  - $num - antal rader i $result.
  - $result - Resultatet av SQL-frågan.

  En funktion som presenterar resultatet av en SQL-fråga i en HTML-tabell.

mysqlconn.php Ansluter till MySQL-databasen.

- *db_escape($post)*
  - $post - Array.

Funktion för att förhindra SQL-injektioner, eller i lindrigare fall MySQL-fel.

noLogin.php Den sidan som användaren kommer till om han/hon inte är anställd på universitetet då användarens NetID inte finns i personalkatalogen.

pdf.php Innehåller funktioner som skapar PDF-filerna med hjälp av `fpdf.php`.

- *create_pdf_all($filename,$netid,$inaktiveras)*
  - $filename - filnamnet på textfilen som data läses ifrån.
  - $netid - NetID på användaren som skapade filen.
  - $inaktiveras - När gästkontot inaktiveras.

  Skapar en PDF-fil med en sammanställning av alla konton som skapats i samma omgång.

- *create_pdf_eng($antal,$filename,$netid,$inaktiveras)*
  - $antal - antalet sidor som ska skrivas i PDF-filen.
  - $filename - filnamnet på textfilen som data läses ifrån.
  - $netid - NetID på användaren som skapade filen.
  - $inaktiveras - När gästkontot inaktiveras.

  Skapar en PDF-fil med så många sidor som det är skapade konton. Varje sida innehåller uppgifter till ett gästkonto. Instruktioner på engslska för hur uppgifterna används finns på varje sida.

- *create_pdf_sv($antal,$filename,$netid,$inaktiveras)*
  - $antal - antalet sidor som ska skrivas i PDF-filen.
  - $filename - filnamnet på textfilen som data läses ifrån.
  - $netid - NetID på användaren som skapade filen.
  - $inaktiveras - När gästkontot inaktiveras.

  Skapar en PDF-fil med så många sidor som det är konton. Varje sida innehåller uppgifter till ett gästkonto. Instruktioner på svenska för hur uppgifterna används finns på varje sida.

- *create_pdf_sv_eng($antal,$filename,$netid,$inaktiveras)*
  - $antal - antalet sidor som ska skrivas i PDF-filen.
  - $filename - filnamnet på textfilen som data läses ifrån.
  - $netid - NetID på användaren som skapade filen.
  - $inaktiveras - När gästkontot inaktiveras.

  Skapar en PDF-fil med så många sidor som det är konton. Varje sida innehåller uppgifter till ett gästkonto. Instruktioner för hur uppgifterna används finns också. Texten är både på engelska och svenska.

reset_pw_all.php Inkluderar `ldapfunctions.php`, `pdf.php`, `checkfunctions.php` och `auth.php`. Det är `reset_pw_all.php` som anropas när användaren trycker på knappen ”Ändra alla lösenord” på `changeAccount.php`. Ett HTML-formulär tas emot från `changeAccount.php` med de gästkonton vars lösenord ska ändras. Ett nytt lösenord genereras och skrivs till gästkontot. Övriga attribut hämtas sedan från kontot för att kunna skriva en post i databasen. De nya lösenorden till respektive gästkonto skrivs ut tillsammans med användarnamnet. PDF-filerna som skapas finns tillgängliga via länkar för användaren.

users.php Inkluderar `ldapfunctions.php`, `checkfunctions.php` och `auth.php`. `users.php` är andra steget när en användare ska aktivera nya gästkonton. Data från `createAccount.php` kommer i ett HTML-formulär och kontrolleras i `users.php` så den datan är korrekt. De kontroller som sker är i ordning följande

- Syftet får inte vara mer än 45 tecken.
- Alla specialtecken tas bort, inkluderat mellanslag, tabbar och så vidare.
- Om datumet är tomt så sätts det till dagens datum plus sju dagar.
- Syftet kontrolleras så det inte tomt.
- Datumet måste vara i korrekt format (YYYY-MM-DD).
- Kontrollerar så datumet inte har passerats.
- Kontrollerar så att datumet är giltligt.
- Kontrollerar så att minst ett namn är angivet, så att minst ett konto kan skapas.
- Separerar namn och e-postadress samt tar bort specialtecken.
- Kontrollerar syntaxen på e-postadressen. I de fall e-postadress saknas sätts den till `null@kau.se`.
- Lösenord genereras i en array.
- Kontrollerar så att datumet ligger max 30 dagar fram i tiden.
- Kontrollerar så att det finns tillräckligt med lediga konton i AD.

Går samtliga kontroller igenom utan fel, så skrivs nu datan ut och läggs i ett dolt HTML-formulär för att kunna skickas till `activateUser.php`.

ldapfunctions.php Här finns alla funktioner som kommunicerar med AD. Denna fil inkluderar `mysql.php`.

- *activate_user($namn, $mail, $ldapconn, $daysactive, $syfte, $netid, $pass, $inaktiveras, $filname)*

  - $namn - Gästens namn.
  - $mail - Gästens e-postadress.
  - $ldapconn - Koppling till LDAP-server.
  - $daysactive - Antal dagar kontot är aktiverat.
  - $syfte - Angivet syfte med kontot.
  - $netid - Användaren som aktiverade kontot.
  - $pass - Lösenordet till kontot.
  - $inaktiveras - Datum då kontot upphör att gälla.
  - $filename - Filnamn på en temporär textfil.

  Returnerar ett kontonamn om aktiveringen lyckas annars falskt. Anropar funktionerna *changepwd()*, *addToLog()*, *createTempTextFile()* och *unixtoad()*.

- *adminsearch($string, $ldapconn)*

  - $string - Sökordet.
  - $ldapconn - Koppling till LDAP-server.

  Används av administratörsdelen. Anropar funktionen *unixtoad*. Söker i OU på attributen `cn, physicaldeliveryofficename` och `description`. Söker med jokertecken samt skriver ut resultatet.

- *adtimetonormal($time)*

  - $time - Tiden med AD tidstämpel.

  Returnerar Unixtid. Räknar ut tid från AD-tidstämpel till Unix-tidstämpel

- *changepwd($username, $username2, $pass)*
  - $username - Kontonamn.
  - $username2 - Kontonamn med komplett domännamn.
  - $pass - Det nya lösenordet.

  Returnerar sant om det lyckas annars falskt. Ändrar lösenord för givet konto. Kontot aktiveras för att lösenordsbytet ska gå att genomföra.

- *countdisableaccounts($ldapconn)*
  - $ldapconn - Koppling till LDAP-server.

  Returnerar antalet lediga konton i OU. Anropar funktionen *unixtoad*

- *createTempTextFile($username, $pass, $displayname, $mail, $inaktiveras, $filename)*
  - $username - Kontonamn.
  - $pass - Lösenordet.
  - $displayname - Gästens namn.
  - $mail - Gästens e-postadress.
  - $inaktiveras - Datum då kontot upphör att gälla.
  - $filename - Filnamnet på .txt-filen.

  Skriver till en .txt-fil som används för att skapa PDF-filer.

- *deactivate_user($username, $ldapconn)*
  - $username - Kontonamn.

  - $ldapconn - Koppling till LDAP-server.

  Returnerar sant om det lyckas annars falskt. Avaktiverar givet konto i OU.

- *edit_user_with_pw($username, $namn, $mail, $ldapconn, $daysactive, $syfte, $netid, $pass, $inaktiveras, $filename)*

  - $username - Kontonamn.
  - $namn - Gästens namn.
  - $mail - Gästens e-postadress.
  - $ldapconn - Koppling till LDAP-server.
  - $daysactive - Antal dagar kontot är aktiverat.
  - $syfte - Angivet syfte med kontot.
  - $netid - Användaren som ändrade kontot.
  - $pass - Lösenordet till kontot.
  - $inaktiveras - Datum då kontot upphör att gälla.
  - $filename - Filnamn på en temporär .txt-fil.

  Returnerar kontonamnet om ändringarna lyckas annars falskt. Ändrar attribut och lösenord för givet konto. Anropar funktionerna *addChangeToLog* och *createTempTextFile*.

- *edit_user_without_pw($username, $namn, $mail, $ldapconn, $daysactive, $syfte, $netid, $inaktiveras, $filename)*

  - $username - Kontonamn.
  - $namn - Gästens namn.
  - $mail - Gästens e-postadress.

- $ldapconn - Koppling till LDAP-server.
- $daysactive - Antal dagar kontot är aktiverat.
- $syfte - Angivet syfte med kontot.
- $netid - Användaren som ändrade kontot.
- $inaktiveras - Datum då kontot upphör att gälla.
- $filename - Filnamn på en temporär .txt-fil.

Returnerar kontonamnet om ändringarna lyckas annars falskt. Ändrar attribut på givet konto. Anropar funktionerna *addChangeToLog* och *createTempTextFile*.

- *getaccountinfo($username, $ldapconn)*
  - $username - Kontonamn.
  - $ldapconn - Koppling till LDAP-server.

  Returnerar en multidimensionell array med kontoattribut. Anropar funktionen *unix-toad*. Hämtar alla attribut från ett givet konto.

- *getDesc($cn, $ldapconn)*
  - $cn - Kontonamn.
  - $ldapconn - Koppling till LDAP-server.

  Returnerar attributet `description` från ett givet konto

- *getexpdate($i, $ldapconn)*
  - $i - Sista delen på ett kontonamn, en siffra.
  - $ldapconn - Koppling till LDAP-server.

  Returnerar en 64-bitars integer. Tiden då ett givet konto upphör att gälla returneras med AD-tidstämpel.

- *getinaktiveras($cn, $ldapconn)*
  - $cn - Kontonamn.
  - $ldapconn - Koppling till LDAP-server.

  Returnerar datumet då ett konto upphör att gälla på formatet ÅÅÅÅ-MM-DD. Anropar funktionen *adtimetonormal*. Gör om från AD- till Unix-tidstämpel och formaterar den.

- *getmail($cn, $ldapconn)*
  - $cn - Kontonamn.
  - $ldapconn - Koppling till LDAP-server.

  Returnerar attributet `mail` från givet konto.

- *getname($cn, ldapconn)*
  - $cn - Kontonamn.
  - $ldapconn - Koppling till LDAP-server.

  Returnerar attributet `displayname` från givet konto.

- *getuseraccount($username, $ldapconn)*
  - $username - NetID på användaren.
  - $ldapconn - Koppling till LDAP-server.

  Returnerar det antal konton ett visst netID har aktiverat och som fortfarande är aktiva. Anropar funktionen *unixtoad*. NetID står i attributet `physicaldeliveryofficename`.

- *getuseraccount_ admin($ldapconn)*
  - $ldapconn - Koppling till LDAP-server.

Returnerar det antal konton som används. Anropar funktionen *unixtoad*.

- *printuseraccount($username, $searchstring, $ldapconn, $type)*
  - $username - Användarens netID.
  - $searchstring - Sökord.
  - $ldapconn - Koppling till LDAP-server.
  - $type - Vilket filter som ska användas.

  Skriver ut kontoinformation efter sökning i OU. Funktionen används på användarsidan. Sökningen sker med olika filter. Anropar funktionen *unixtoad*.

- *setExpDate($numdays)*
  - $numdays - Antalet dagar.

  Returnerar tid med AD-tidstämpel. Räknar om ett antal dagar till AD-tid från och med dagens datum.

- *totnumaccount($ldapconn)*
  - $ldapconn - Koppling till LDAP-server.

  Returnerar det antal konton som finns i OU. Räknar alla konton både aktiva och inaktiva.

- *unixtoad()*
  Returnerar en 64-bitars integer. Räknar om nuvarande Unix-tid till AD-tidstämpel.

accountListAdmin.php Inkluderar *auth.php* och *ldapfunktions.php*. En administratörsida som ger administratören möjlighet att söka och avaktivera konton i AD.

activateUser.php Inkluderar *auth.php, ldapfunctions.php, checkfunctions* och *pdf.php*. En användarsida som visar resultatet av aktivering av konton. Kontonamn med respektive lösenord och gästanvändarnamn visas. Kontouppgifter hämtas som POST-variabler. Konton aktiveras. E-post skickas till de gästanvändare som har giltig e-postadress. PDF-filerna skapas och skickas till användarens netID. Den textfil som används för att skapa PDF-filerna töms.

auth.php Inkluderar *kau_api_service_call.php* och *CAS.php*. Initierar phpCAS genom att ange version, CAS-server och portnummer. Användaren tvingas logga in. Utloggning från CAS och kontroll av netID hanteras här.

changeAccount.php Inkluderar *auth.php* och *ldapfunctions.php*. En användarsida där användarens aktiva konton listas. Användaren kan söka på olika attribut bland "sina" aktiva konton. Användaren kan ändra attribut och avaktivera konton. GET-variabler kontrolleras för att meddela om en ändring lyckats eller inte.

createAccount.php Inkluderar *auth.php* och *ldapfunctions.php*. En användarsida med HTML-formulär för att ange den information som behövs för att aktivera konton. GET-variabler kontrolleras för att meddela om någon information är felaktig efter kontroll av informationen.

delete.php Inkluderar *auth.php* och *ldapfunctions.php*. Avaktiverar ett konto i OU. Kontrollerar om det är en användare eller administratör som avaktivera det och skriver denna information till loggdatabasen. Sätter GET-variabler för lyckad eller misslyckad avaktivering.

delete_all.php Inkluderar *auth.php* och *ldapfunctions.php*. Avaktiverar $antal konton i OU. Kontrollerar om det är en användare eller administratör som avaktivera det och skriver

denna information till loggdatabasen. Sätter GET-variabler för lyckad eller misslyckad avaktivering.