Satakunnan ammattikorkeakoulu

Ville Aho

PALVELINYMPÄRISTÖN LUONTI MOBIILISTREAMING PALVELULLE

Tietotekniikan koulutusohjelma
Tietoliikenteen suuntautumisvaihtoehto

2007

PALVELINYMPÄRISTÖN LUONTI MOBIILISTREAMING PALVELULLE

Aho, Ville Henrikki
Satakunnan Ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Kesäkuu 2007
Ohjaaja: Niemi, Juha DI
UDK: 621.395:621.371, 004.4 ja 004.6
Sivumäärä: 72

Avainsanat: palvelimet, Linux, PHP, MySQL, streaming

Palvelinympäristön luonti oli toinen osa projektia, jossa luotiin lisäksi ohjelmisto matkapuhelimeen, joka osasi toistaa musiikkia palvelimelta tallentamatta sitä. Tavoitteenamme oli luoda helposti muokattava palvelinympäristö, joka olisi samalla käyttäjäystävällinen. Alkuperäisenä suunnitelmanamme oli siirtää musiikkia mp3-muodossa http:n yli. Palvelinympäristö luotiin syksyllä 2006

Palvelinympäristön teoriaosuudessa käsiteltiin streaming-tekniikkaa yleisesti, mobiililaitteiden streamingia ja palvelimia.

Toteutuksen yhteydessä käytimme tietojamme streaming-tekniikoista ja palvelimista lopullisen tuloksen aikaansaamiseksi. Halusimme luoda ympäristön, joka oli helposti muokattavissa mahdollisten ongelmien kiertämiseksi.

Työ aloitettiin asentamalla Mandriva 2006 Free käyttöjärjestelmäksi. Järjestelmään asennettiin www-palvelimeksi Apache. Tietokannaksi asensimme MySQL:n ja www-sivujen dynaamisuudeksi PHP:n.

Käyttäjien luontia varten tehtiin oma sivusto, johon pääsi vain ennalta määritellyllä salasanalla. Tunnukset sijoitettiin tietokantaan ja lisäksi sftp:tä varten käyttöjärjestelmään.

Palvelun käyttäjä pääsi omille sivuilleen luomillaan tunnuksilla ja kirjauduttaessa käyttäjän IP-osoite tallennettiin tietokantaa. Toiminnoiksi käyttäjille annettiin mahdollisuuksia luoda, muokata ja poistaa soittolistoja. Jälkeenpäin lisäsimme kappaleiden poiston ja siirron verkkohakemistoon. Siirtotoiminto tarvittiin, koska rajoitimme käyttäjän pääsyä public_html-kansioon.

Nykyisten matkapuhelinten rajoittuneisuuden takia päätimme kokeilla erillisiä streaming-palvelimia. Muutos edellytti musiikin tiedostotyypin muuttamista amr:ksi. Kokeilemamme vaihtoehdot olivat Flash Media Server2, Darwin Streaming Server ja Live555. Ainoastaan Live555 sisälsi haluamamme ominaisuudet. Testattuamme matkapuhelimella soittaa musiikkia Live555:n avulla havaitsimme epäluotettavuutta soitossa ja päätimme siirtyä takaisin alkuperäisiin suunnitelmiin.

Havaittuamme uusien S60 3rd fp1 puhelimien tukevan streamausta http:n päällä amr-tiedostoille, päätimme vaihtaa vain tiedostomuotoa.

SERVER ENVIRONMENT FOR A MOBILE STREAMING SERVICE

Aho, Ville Henrikki
Satakunta University of Applied Sciences
Degree Programme in Information Technology
June 2007
Lecturer: Niemi, Juha MSc
UCK: 621.395:621.371, 004.4 and 004.6
Number of pages: 72

Key words: servers, Linux, PHP, MySQL, streaming

Creating a server environment was the other part of the project where also software for mobile phones was created which would be able to reproduce music from the server without storing it first. The goal was to create an easily adjustable server environment that would also be user friendly. The original plan was to transfer music in the mp3 format over the http. The server environment was created in the autumn 2006.

In the theory section of the server environment that was covered, the topics were streaming techniques in common, streaming with mobile devices and servers.

While we were working among the realization we applied our knowledge of streaming techniques and servers in order to create final product. We wanted to create environment that was easily modifiable for making workarounds for possible problems.

Work was started with the Mandriva 2006 Free installation as the operating system. Apache was installed as the web server into the system. MySQL was selected to be installed as the database and PHP to provide dynamics to the web pages.

The web page was made for the creation of users where there was access only with the password that was made earlier. Accounts were positioned into the database and also into the system for sftp access.

The user of the service was able to access his own pages with keys that were made before and when logging the user's IP address was stored into the database. We decided to give the user the ability to create, modify and remove playlists. Afterwards we added abilities to remove songs and to move them into web folder. Moving of the songs was needed because we limited users' access to a public_html folder.

Because of the limitations of present mobile phones we decided to try separate streaming servers. Modification required changing the file format as amr. Options that we tried were Flash Media Server2, Darwin Streaming Server and Live555. Only Live555 included the properties that we wanted. After having tested playing music with the help of Live555 we noticed unreliability in playing and decided to move back to original plans.

We didn't decide to change the file format until we noticed that the new S60 3rd fp1 mobile phones support streaming over the http to amr files

ESIPUHE

Tämä projekti sai alkunsa aivan sattumalta. Kun radioasemilta ei tullut mitään kuunneltavaa musiikkia, niin minulle ja Antti Suomiselle tuli mieleen ajatus matkapuhelimen välityksellä toistettavasta musiikista. Idean omaperäisyys kiehtoi, sillä vuonna 2005 matkapuhelinten musiikkiominaisuuksien hyötykäyttö oli vielä ottamassa ensimmäisiä askeleitaan. Kovin pitkää aikaa ei mennyt idean syntymisestä sen muovaamiseen opinnäytetyöksi sopivaksi kokonaisuudeksi kahdelle henkilölle. Jaottelu kahteen melko tasaiseen osaan onnistui helposti, sillä oli selvää, että tarvittiin asiakas-palvelin -mallin mukainen vuorovaikutus. Varsinaisessa toteutuksessa kahtiajakoa ei kuitenkaan ollut ja työtä tehtiin parina niin palvelinpuolella kuin matkapuhelimen sovelluksen parissakin. Työtahti oli välillä melko raju ja vuorokausista tuntui loppuvan tunnit kesken. Missään vaiheessa ei kuitenkaan tullut mieleen luovuttaa, vaikka välillä jotkin ongelmat tuntuivat ylitsepääsemättömiltä.

Haluan erityisesti kiittää Antti Suomista, sillä tämä työ olisi tuskin yhtä mielenkiintoinen ilman osallisuuttasi ja kiitän myös panostuksestasi tähän projektiin. Haluan myös kiittää ohjaajaani Juha Niemeä sekä Ismo Trastia, Sami Peltomäkeä ja Reino Aarista, jotka osaltaan auttoivat ja neuvoivat työn tekemisessä.

SISÄLLYS

1 JOHDANTO

Tämä opinnäytetyö soveltuu hyvin niille, jotka suunnittelevat jonkinlaisen interaktiivisen verkkopohjaisen palvelun suunnittelua ja jotka eivät ole aivan varmoja sen toteutustavasta tai siitä millaisia rajoituksia voi tulla vastaan ja miten niitä voi ratkaista. Ennen kaikkea, mitä tulee ottaa huomioon ennen kuin aloittaa varsinaisen työvaiheen.

Opinnäytetyön perimmäinen tarkoitus oli luoda tavalliselle käyttäjälle palvelu, jonka avulla voisi kuunnella musiikkia matkapuhelimella. Tarkoituksena ei ole, että käyttäjä lataisi kappaleet puhelimeensa. Sen sijaan ne olisi tallennettuna palvelimelle, josta sitä lähetetään käyttäjälle kappaleen etenemisen mukaan.

Idea kyseiseen palveluun syntyi käytännön tarpeesta omaan, helposti toistettavaan musiikkiin, joka kuitenkin liikkuisi käyttäjän mukana. Hieman ennen idean syntymistä oli otettu käyttöön ensimmäiset mobiili-laajakaistat ja markkinoille tuli myös mp3-formaattia tukevia puhelimia. Toisin sanoen oli tullut mahdolliseksi teknisesti toteuttaa palvelu rajatulla alueella. Muutosta nopeampiin mobiiliverkkoihin hidastaa edelleen se, että matkapuhelinverkon datasiirron nopeuttaminen tarkoittaa tukiasemien välisten etäisyyksien lyhentämistä sekä niiden kapasiteetin kasvattamista.

Aloitushetkellä puhelimien muistit olivat vielä hieman pienempiä kuin nykyiset ja huomattavasti kalliimpia. Siksi ajatus muualle kuin puhelimeen tallennetusta musiikista vaikutti käytännöllisemmältä. Lisäksi mikäli tämän tyyppisiä palveluita tulisi enemmän markkinoille, se saattaisi lisäksi nopeuttaa nopeampien mobiiliverkkojen leviämistä. Palvelun tyypillä tarkoitan langattomia palveluita, jotka tarvitsevat tai hyötyvät nopeista ja kiinteähintaisista yhteyksistä.

Alussa suunnittelimme projektin pääpiirteittään seuraavanlaisiin osiin:

1. WWW-palvelimen asennus
2. Käyttäjätunnuksenluonti

3. Käyttäjän oma sivusto ja sisältö
4. Sisällön esittäminen ja sen muokkaaminen

5. Yksinkertaisen sovelluksen rakentaminen puhelimeen, joka kykenee toistamaan ja lataamaan musiikkia samanaikaisesti
6. Lopullisen puhelimen sovelluksen rakentaminen, joka kykenee tarkastelemaan erilaisia soittolistoja

Yllä oleva esitys on hyvin karkea suunnitelma projektiin. Aikaisemman kokemuksen puuttuessa oli tehtävä suurpiiteisiä linjauksia ja edettävä kulloisenkin tilanteen edellyttämällä tavalla.

Työssä edettiin muutaman kerran siihen tilanteeseen, jossa havaittiin, että kyseisellä menetelmällä ei saada halutunlaista tulosta ja täytyy palata hieman taaksepäin. Aina kuitenkin löytyi jonkinlainen ratkaisu ongelmiin, jolla päästiin taas jatkamaan eteenpäin ja samalla saatettiin havaita uusia ja parempia ratkaisuja kuin aiempi toteutustapa olisi tuottanut.

2 TEORIATAUSTAA

Työn tekemiseksi tarvitsemamme teoriatietous sisälsi tietoa streaming-tekniikoiden käyttöstä , streamingistä mobiililaitteiden kanssa ja palvelinten toiminnasta. Tutkimme teorioita jo osittain tietoisina valinnoista. Näitä valintoja olivat Linuxin käyttö palvelimen alustana, puhelimen käyttöjärjestelmänä Symbian Series 60 3rd edition ja tekniikka, jolla musiikki toistetaan latauksen aikaisesti. Osa-alueet on käsitelty siinä määrin kuin itse työn tekeminen edellytti.

2.1 Palvelimet

Palvelimella käsitetään ohjelmisto, jolla on kyky vastata ulkoisiin palvelupyyntöihin. Palvelimen tyyppi riippuu sen kyvystä vastata tietynlaisiin pyyntöihin. Esimerkiksi web-palvelin osaa vastata http:n kautta tulleisiin pyyntöihin. Usein palvelinkoneen rakentaminen lähtee sen mitoituksesta tarpeen mukaan. Toisin sano-

en arvioidaan kuinka paljon tarvitaan tehoa ja muistia tietyn käyttäjämäärän palveluun. Seuraavana vaiheena on palvelinohjelmistojen valinta, sillä palvelin ilman palvelinohjelmistoja on tavallinen työasema. /6, 7/

2.1.1 Apache

Palvelinohjelmistojen valintaan vaikuttaa luonnollisesti käytön kohde ja käyttäjien määrä. Ensimmäinen palvelinohjelmisto oli Apache, joka on yleisesti suosituimpia http-palvelimia. Sen suosio perustuu tavallisen käytön helppouteen ja tuesta monille eri alustoille. /7/

Apache itsessään ei ole kovin laaja ja tavallisena asennuksena se sisältääkin vain toiminnat, joilla voidaan siirtää tiedostoja http:n yli. Suurin etu siinä on kuitenkin sen laajennettavuus moduuleilla. Erilaisia lisämoduuleita on monenlaisia ja esimerkiksi käyttämäämme versioon 2.0 oli noin 55 kappaletta. Niiden toiminnot olivat hyvin vaihtelevia. Joidenkin moduulien tehtävänä on mahdollistaa käyttäjien autentikointi, kun taas toiset moduulit toteuttivat virtuaalista isännöintiä (*virtual hosting*). Apache:n suurimpia etuja ovat moni prosessointi moduulit (MPM). Niiden tehtävänä on luoda yhteys koneen verkkoportteihin, hyväksyä pyyntöjä ja määrittää palveleva moduuli. Ilman erillistä määrittelyä MPM määräytyy käyttöjärjestelmän mukaan. Linuxissa oletuksena on *prefork* MPM, joka on hyvä järjestelmissä, joiden tarkoituksena on pitää pyynnöt erillään toisista käyttäjistä. Tämän tyylinen ratkaisu on erittäin käytännöllinen, kun haetaan luotettavuutta ja yhteensopivuutta, sillä virheen tapahtuessa yksittäiselle käyttäjälle, se ei vaikuta samalla muihin. /1, 4, 5/

Palvelinohjelmiston toimintaa muutetaan toimintaohjeilla (directives). Kyseiset ohjeet sijaitsevat Mandriva 2006 Free:ssä tiedostossa */etc/httpd/conf/httpd.conf*. Kyseisessä tiedostossa määritetään ladattavat ekstramoduulit ja poikkeavat sivustojen tiedostopäätteet. Ladattavat moduulit on sijoitettava */etc/httpd/modules.*d-kansioon omina hakemistoinaan. Moduulin hakemisto sisältää asetustiedoston.

Kyseinen asetustiedosto sisältää tekstimuodossa moduulin toiminnan. Tiedosto on puhtaasti tekstipohjainen, joten sen muokkaamiseen ei tarvitse mitään erikoistyökaluja, vaan pelkän tekstieditorin. Tiedostoa muokattaessa on kuitenkin otettava huomioon, että kirjoitusvirheet saattavat johtaa http-palvelimen huonoon toimintaan tai toimimattomuuteen. Tehdyt muutokset tulevat voimaan, kun Apachen palvelu käynnistetään uudelleen (httpd). Uudelleenkäynnistyksen tarve johtuu siitä, että httpd kääntää asetustiedoston käynnistyessään. /2/

2.1.2 MySQL

MySQL on tulkki tietokannan ja käyttäjän välillä. Se osaa suorittaa tietokantakyselyitä käyttäjälle helpommasta muodosta ja tehdä siitä tietokantaan soveltuvan monimutkaisen kyselyn. MySQL on relaatiopohjainen tietokannan hallintajärjestelmä. Arkisemmassa kielessä sitä sanotaan myös relaatiotietokannaksi, sillä sen toiminnat järjestelmän kanssa ovat näkymättömiä käyttäjälle. Suurimpina etuina ovat sen ilmaisuus ja avoimuus. Nämä asiat ovat mahdollistaneet monien työkalujen kehityksen sekä helpottaneet siinä ilmenneiden vikojen korjaamisessa, mikä ilmenee sen luotettavuutena ja nopeutena. /11/

Palvelinohjelmistolla on mahdollisuus käyttää erilaisia tietokantamoottoreita. Käytössämme olleessa MySQL:n versiossa 4.1 oli vielä käytössä *MyISAM* -niminen moottori. MySQL:n versiossa 5 ja siitä eteenpäin ei ole enää MyISAM:a. Tieto tauluista tallennetaan kolmeen eri tiedostoon. Taulun muoto sisällytetään *frm*-tiedostoon. Tieto, joka on tallennettuna tauluun, on *MYD*-päätteisessä tiedostossa ja indeksi- tiedosto on *MYI*-päätteisessä tiedostossa. Esimerkiksi luotaessa users -niminen taulu MyISAM:a käyttäen siitä muodostuisi tiedostot users.frm, users.MYD ja users.MYI. MyISAM tukee 63 bittisiä tiedostoja. Taulu voi sisältää 2^{32} riviä tai nostamalla rajoitusta asennettaessa käyttämällä *–with-big-tables* -valintaa saa tauluun $(2^{32})^2$ riviä. Sarakkeita voi olla enintään 16 kappaletta indeksiä kohden. /1/

InnoDB tuo mukanaan muutamia parannuksia MyISAM:in nähden. Etuina ovat uudet varmuusmenetelmät, jotka koskevat tapahtumaturvaa. Nämä tapahtumakohtaiset turvamenetelmät käsittävät kuorman, takaisinkierron ja kaatumisesta palautumisen kaltaiset ominaisuudet. InnoDB:n ehkä merkittävin etu on mahdollisuus suorittaa hakuja ulkoisen avaimen perusteella myös muunlaisista kuin InnoDB tietokannoista. Tietokantamoottori on suunniteltu huolehtimaan suurista tietomääristä ja käsittelemään monia kyselyitä samanaikaisesti. /11/

2.2 Streaming

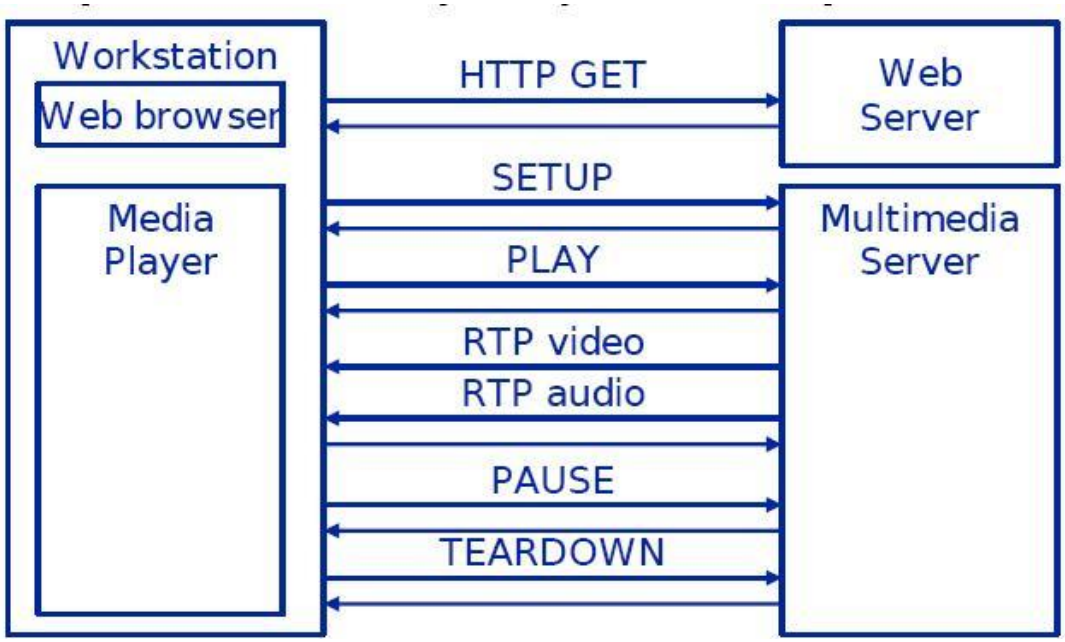
On olemassa kahden tyyppistä streamausta (suoratoistoa): sellaista jossa tietoa ei varastoida etukäteen ja sellaista, jossa tietoa varastoidaan etukäteen enemmän kuin toistamiseen vaaditaan. Tämä osio on jaettu kahteen osaan: suoratoiston yleisiin periaatteisiin ja sen toimintaan matkapuhelimissa.

2.2.1 Streamingin yleiset periaatteet

Streamingina tunnettu tekniikka otettiin käyttöön vuonna 1995. Englanninkielessä streaming ei välttämättä tarkoita samaa kuin suomenkielessä sana streamaus. Englanninkielisessä versiossa usein tarkoitetaan vain tiedon yhtenäistä siirtoa, eikä niinkään siirronaikaista toistoa kuten suomenkielisellä streamauksella usein käsitetään. Streameista puhuttaessa on hyvä varmistua sen käyttötarkoituksesta ja käytetyistä menetelmistä. Tapoja käyttää streamingia ovat On-Demand ja Broadcasting tai live. On-demand on Unicastia ja sitä toteutetaan vain erillisestä pyynnöstä. Broadcast on multicastia, eikä käyttäjä pysty vaikuttamaan saamaansa ääneen tai videoon. Tässä tilanteessa en keskity broadcast tyyppiseen streamaukseen, sillä kyseisellä menetelmällä ei ollut käyttöä työssämme. /8/

Yleisin keino lähettää stream:a on käyttää RTP:tä ja RTSP:tä, jotka ovat teollisuusstandardeja. RTSP (Real Time Streaming Protocol) on asiakkaan ja palvelimen välissä oleva kontrollointi ja viestintäprotokolla. Palvelu käyttää porttia 554. Varsinainen tietosisältö voidaan lähettää joko UDP:llä tai TCP:llä. Yleisesti suosi-

taan UDP:tä, sillä se ei kärsi yhtä paljon virheistä linjalla kuten TCP. RTSP:n ohjausviestejä ovat *describe*, *setup*, *play*, *pause*, *record* ja *teardown*(kuva 1). Esimerkiksi *describe*-sanomaa käytetään kertomaan toistettavan tiedoston sijainti muodossa *rtsp://palvelin/kappale.mp3*.



Kuva 1. RTSP-viestit /12, s. 131/

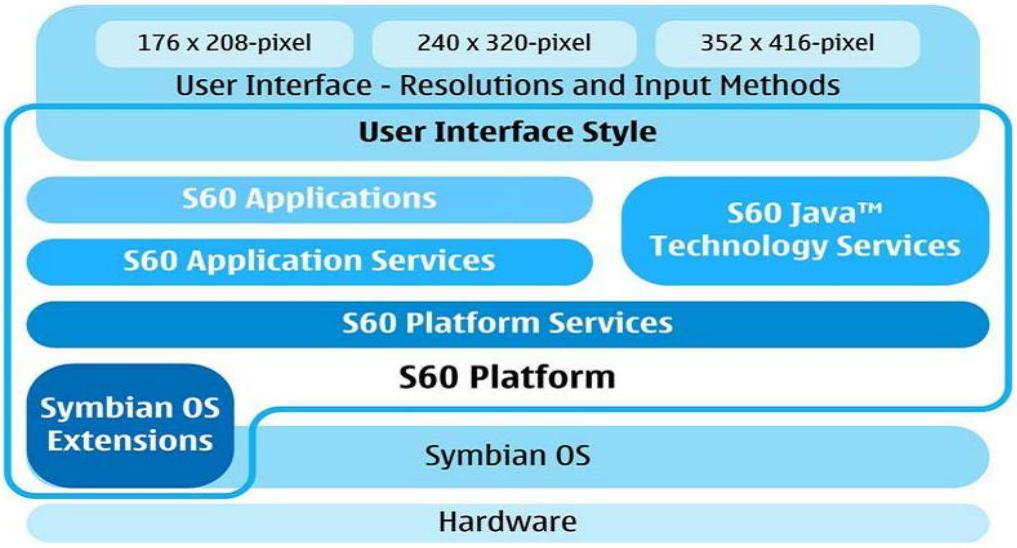
RTP:tä (Real-time Transport Protocol) käytetään palvelimella lähettämään asiakkaalle tietoa asiakkaalle. Kyseinen protokolla toistaa tiedon reaaliajassa pelkän siirron sijaan. /12, 8/

RTMP (Real Time Messaging Protocol) on alun perin Macromedian kehittämä protokolla. Sen pääasiallinen käyttökohde on Flash Media Server:n ja Flash playerin välinen streami. Protokolla ei lähetä viestejä asiakkaan ja palvelimen välillä, vaan ylläpitää pysyvää yhteyttä kahden päätepisteen välillä. RPC:t (Remote Procedure Call) toteutetaan asynkronisesti yksittäiselle palvelimelle tai työasemalle pyynnöillä ja vastauksilla. Protokollasta on tehty joitakin muunnelmia kuten RTMPT, joka tarkoittaa http:n kautta tunneloitua yhteyttä. Tunnelointi helpottaa toimintaa palomuurien kanssa. /19/

Aiemmista poikkeava tapa toteuttaa streamaus on käyttää puhtaasti http:tä. Käytettäessä pelkkää http:tä täytyy asiakasohjelman osata koostaa saamistaan palasista toistettava osuus. Tällöin myös tieto tallennetaan väliaikaiseen muistiin. Palvelimen kannalta http:n kautta tapahtuva streamaus on helpoin, koska tieto välittyy asiakasohjelmalle vastauksessa. Etuna http-streamauksessa on myös palomuurisäännösten helppous. Usein http-streamauksesta käytetään myös nimitystä progressiivinen streami. /8/

2.2.2 Mobiililaitteiden Streaming

Puhuttaessa rajoittuneiden kannettavien laitteiden streamauksesta ei voida tehdä mitään yleispäteviä linjauksia, sillä niiden tukemat tiedostomuodot ja protokollat vaihtelevat suuresti. Edes saman valmistajan laitteet eivät noudata samoja protokollia. Tuettujen tiedostomuotojen ja protokollien tietäminen vaatii tutustumista puhelimen tai muun mobiililaitteen alustaan. Esimerkiksi käyttämämme puhelimen käyttöjärjestelmä oli S60 3rd ja sen rakenne oli kuvan 2 mukainen:



Kuva 2. N80:n rakenne kerroksittain /9, s. 10/

Kuten kuvasta 2 voidaan havaita, Javalla ei pääse suoraan käsiksi laitteistoon. Toiminnot, jotka Javalla ovat käytössä, periytyvät sitä palvelevalta kerrokselta

(Platform services). Tätä taustaa vasten on hyvä lähteä tutkimaan todellisia mahdollisuuksia streamauksessa. /9, 10/

Käytettäessä RTSP:tä ainakaan Nokian puhelimissa ei ole suoraa tukea sille, sillä ne käyttävät RealPlayerin APN:ää. Tämä tarkoittaa, että ulkoinen ohjelma hoitaa RTSP:n käsittelyn. Ainoa tapa tällöin varmistua RTSP:n kunnollisesta toiminnasta on tarkistaa RealPlayerin yhteysasetukset. /9, 10/

Progressiivisen streamauksen (http streaming) tapauksessa täytyy varmistua puhelimen tuesta sille. Toinen huomioitava asia kyseisessä yhteydessä on tarkistaa progressiivisen streamauksen tukemat tiedostomuodot. Esimerkkinä mainittakoon käyttämästämme mallista (S60 3rd fp1), että se ei tukenut progressiivista streamia. S60 3rd fp1 taasen tukee progressiivista streamia amr-tiedostoille. /9, 10/

3 SUUNNITTELU JA ESIVALMISTELU

Alla olevat suunnitelmat ja pohdinnat olivat osa prosessia, joka johti vasta lopullisiin suunnitelmiin. Monesti tuli sellaisia esteitä vastaan, johon ei ollut varauduttu, joko sen takia, että kokonaisuuden hahmottaminen oli välillä melko hankalaa tai joillain toteutuksilla ei olisi päästy lähellekään tyydyttävää tulosta. Yksittäisiä osia ei siis kannata poimia ellei ole varma, että se soveltuu omaan käyttöön.

3.1 Esivalmistelu ja perehtyminen toteutuksen mahdollisuuksiin

Tämän työn suunnittelu käynnistettiin tutkimalla mahdollisuuksia ylipäätään toteuttaa idea. Sen asian selvittämiseksi oli tutkittava eri puhelinmallien ominaisuuksia ja mahdollisuuksia. Päätimme erityisesti tutkia mahdollisuutta toteuttaa palvelu Nokian N80 puhelimella, joka oli käytössämme. Tärkein ominaisuus, joka

puhelimella tuli olla, oli tuki mp3-formaatille. Tämä siis siksi, että mp3 on ehkä yleisin ja tuetuin muoto. Kaiken lisäksi mp3 on pakattu suhteellisen hyvin, joten se ei vie liikaa tilaa ja siirrossa se säästää kaistaa. Puhelimen oli lisäksi tuettava EDGE-tekniikkaa (EGPRS), joka mahdollistaa käytännön maksiminopeudellaan suhteellisen hyvälaatuisen mp3-kappaleen kuuntelun tallentamatta sitä puhelimelle.

Vanhemmissa puhelimissa ohjelmiston kehitys oli aina puhelinkohtaista, koska ei ollut olemassa yleistä tapaa toteuttaa ohjelmia. Meidän valittavanamme oli kaksi vaihtoehtoa: Symbian tai Java. Näistä valitsimme Javan, koska myös Symbian on riippuvainen osittain puhelimen Symbianin sarjasta. Java taas sopii kaikkiin niihin puhelimiin, joissa on kyseinen alusta.

Seuraavana oli vuorossa palvelimen valinta. Tarkoituksiimme sopivimpana pidimme ehdottomasti Linuxia, sillä sen muuntaminen yleishyödylliseksi palvelimeksi on huomattavasti mutkattomampaa kuin esimerkiksi Windows-pohjaisen. Lyhyen tutkinnan ja muutaman kokeilun jälkeen päädyimme käyttämään Mandriva 2006 free:tä, sillä se vaikutti helposti lähestyttävältä. Tarve helposti lähestyttävään Linuxiin johtui siitä yksinkertaisesta asiasta, että aiempaa kokemusta Linux-maailmasta ei juuri ollut. Perusteltua oli siis päätyä sellaiseen vaihtoehtoon, joka ei vaatinut suurta asiantuntemusta peruskäyttöön, aluksi.

3.2 Resurssien hankinta ja niihin perehtyminen

Seuraava luonnollinen siirtymävaihe oli etsiä palvelinta tai pikemminkin alustaa sille. Pienen kyselyn jälkeen saimmekin koululta palvelimelle paikan VMware server-ohjelmistoon. Kyseinen järjestely tarkoitti sitä, että varsinaista laitetta ei ollut olemassa, vaan olimme yksi monista, jotka käyttivät fyysisen laitteen resursseja. Mandrivan hyviin puoliin voi laskea sen pienen tarpeen suorituskyvylle. Esimerkiksi se toimii graafisenakin vain 64MB:n keskusmuistimäärällä ja 32MB:n tekstipohjaisena. Edellä mainitut arvot ovat peräisin Mandrivan omilta

sivuilta: http://www.mandriva.org. Kaikki oli siis valmiina käyttöjärjestelmän asennukseen.

3.3 Käyttöjärjestelmän asentaminen

Käyttöjärjestelmän asentaminen virtuaalisessa ympäristössä tapahtuu lähes samalla tavalla kuin normaalisti. Erona on se, että varsinaista levyä ei voi koneeseen laittaa, vaan tulee käyttää alustalla olevaa imagea, joka sitten mountataan virtuaalisen palvelimen virtuaaliseen asemaan. Tämän jälkeen asennus etenee kuten todellisillakin laitteilla.

Mandrivan asentaminen on helppoa myös sellaisille henkilöille, joilla ei ole kokemusta tekstipohjaisista ympäristöistä. Koko asennus on täysin graafinen. Asennus alkaa kahdella vaihtoehdolla: asennuksella tai lisätiedoilla. Kun valitsee asennuksen, tulee kielenvalintoja ja lisenssisopimuksen hyväksymistä ja muuta tavallista. Näiden jälkeen tulee kohta, joka tulee erityisesti huomioida: turvallisuusasetukset. Mikäli valitsee tasoksi *paranoid* (vainoharhainen), tulee samalla myös määritellä käyttäjä tai ei pääse kirjautumaan palvelimelle.

3.3.1 Levyn osiointi

Turvamääritysten jälkeen tulee levyn partitioinnin vuoro. Valitsimme mukautetun jaon, sillä muuten koko levy olisi ollut yhtä osiota. Osiointi tapahtuu helposti valitsemalla vaalealla merkityn osioimattoman tilan ja tämän jälkeen painamalla create (luo). Ruudussa näkyy säädettävä kokoliuska, osion tyyppi ja mounttauspiste eli se, millä nimellä se myöhemmin ilmenee. Tehty jako näkyy kuvassa 3.

```
[root@mstream /]# cat /etc/fstab
# This file is edited by fstab-sync - see 'man fstab-sync' for details
/dev/sda1 / reiserfs notail 1 1
/dev/sda8 /home ext3 defaults 1 2
/dev/hdc /mnt/cdrom auto umask=0022,user,iocharset=iso8859-15,codepage=850,noaut
o,ro,exec,users 0 0
none /proc proc defaults 0 0
/dev/sda9 /tmp ext3 defaults 1 2
/dev/sda6 /usr ext3 defaults 1 2
/dev/sda7 /var ext3 defaults 1 2
/dev/sda5 swap swap defaults 0 0
[root@mstream /]# fdisk -l

Disk /dev/sda: 12.8 GB, 12884901888 bytes
255 heads, 63 sectors/track, 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1         536     4305388+  83  Linux
/dev/sda2             537        1566     8273475    5  Extended
/dev/sda5             537         604      546178+  82  Linux swap / Solaris
/dev/sda6             605        1260     5269288+  83  Linux
/dev/sda7            1261        1376      931738+  83  Linux
/dev/sda8            1377        1546     1365493+  83  Linux
/dev/sda9            1547        1566      160618+  83  Linux
[root@mstream /]# df
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             4.2G  108M  4.1G   3% /
/dev/sda8             1.3G  154M  1.2G  12% /home
/dev/sda9             152M  4.1M  141M   3% /tmp
/dev/sda6             5.0G  1.1G  3.7G  23% /usr
/dev/sda7             896M  243M  608M  29% /var
[root@mstream /]#
```

Kuva 3. Levyn osiointi

Kuvasta 3 selviää olennaiset asiat levytilan jaosta ja niissä käytetyistä tiedostojärjestelmistä.

Linux:ssa varsinaiset levyt on nimetty sda, sdb, sdc jne. Osion tunnus on levyn nimen perässä oleva numero. Tämä tarkoittaa, että meillä oli yksi fyysinen(virtuaalinen) levy käytössä. Levy oli jaettu viiteen varsinaiseen osioon ja lisäksi oli määritelty swap - levy. Swap levylle oli annettu tilaa 500Mt, mikä riittää varmasti järjestelmän sujuvaan toimimiseen. Varsinaisille osioille jaettiin levytilaa lähinnä satunnaisesti, sillä mitään suuria ohjelmistoja tai useita käyttäjiä ei otettu huomioon. Kyse oli siis pienimuotoisesta testipalvelimesta.

Käytön ja projektin kannalta tärkein osio oli *home*, jonne tuli käyttäjien omat hakemistot. Partitioinnin lopuksi vahvistetaan luotavat osiot, jonka jälkeen varsinaiset osiot syntyvät. Käyttäjien omien hakemistojen käytöstä lisää myöhemmissä kappaleissa.

3.3.2 Lisäohjelmat

Tämän jälkeen asennus jatkui valitsemalla lisäohjelmia. Kyseiset ohjelmat saa jälkeenpäin myös asennettua, mutta vaiva asennuksen yhteydessä on hieman pienempi kuin erikseen asennettuna. Toinen huomattava etu on, että palvelinohjelmistot, kuten apache, käynnistyy automaattisesti tämän jälkeen.

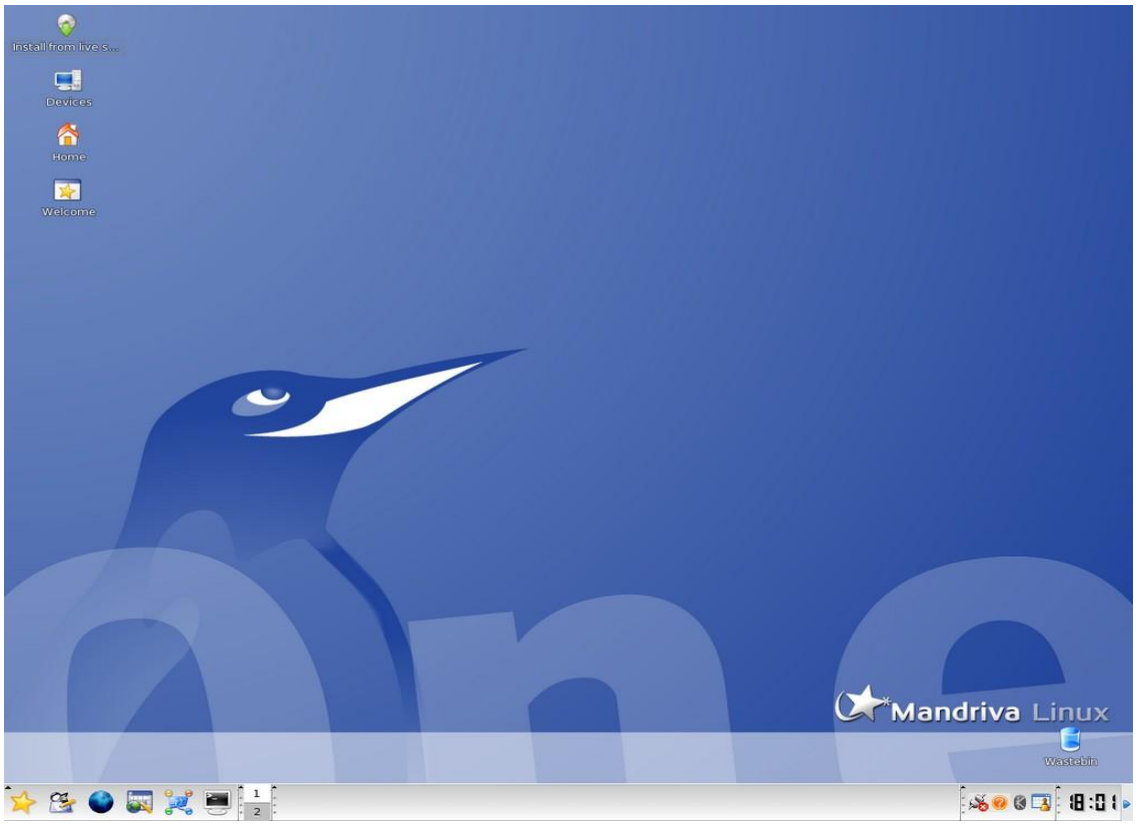
Valitsimme apache:n http – palvelimeksi, OpenSSH:n helpottamaan etähallintaa ja KDE graafiseksi käyttöliittymäksi helpottamaan siirtymistä täysin uudenlaiseen ympäristöön. Apachen konfiguraatioista ja lisämoduuleista selvitetään enemmän kappaleessa 2.1.1.

3.3.3 Asennuksen viimeistely

Ohjelmistojen valinnan jälkeen käyttöjärjestelmä ja ylimääräiset ohjelmat asentuvat. Seuraavana vaiheena ovat järjestelmän asennuksen viimeistely määrittelemällä järjestelmän asetukset. Tärkeimpiä määrityksiä on pääkäyttäjän eli root:n salasanan asettaminen. Tätä kohtaa ei missään nimessä kannata jättää tyhjäksi, sillä silloin kuka vain voi halutessaan tunkeutua järjestelmään ja tehdä haluamiaan muutoksia, kuten asettaa pääkäyttäjälle salasanan, jolloin varsinainen hallinnoija ei pääse sisään tai pahimmassa tapauksessa poistaa kaikki tiedot. Melkein yhtä tärkeä asia on luoda tavallinen käyttäjä, sillä root:na kirjautuminen on hyvin riskialtista ulkoisille hyökkäyksille. Vähemmän merkitseviä asioita ovat järjestelmän kieliasetusten ja aika-asetusten muuttaminen. Nämä tekevät käytöstä hieman helpompaa ja nopeampaa. Lisäksi on hyvä antaa verkkomääritykset ja koska kysees-

sä on palvelin, niin staattinen IP-osoite on tarpeen. Lopuksi tulee enää kysymys ”tallennetaanko asetukset” ja järjestelmä uudelleenkäynnistys.

Onnistuneen asennuksen jälkeen Mandriva käynnistyy ja pyytää käyttäjältä tunnusta ja salasanaa. Kuten jo aiemmin mainitsin, root-käyttäjänä ei kannata kirjautua sisään lukuisten tietoturvariskien takia ja siksi on hyvä jo asennusvaiheessa luoda tavallinen käyttäjä. Kirjautumisen jälkeen aukeaa Mandrivan työpöytä. Asennus on siis suoritettu onnistuneesti ja nyt on olemassa valmis pohja palvelun kehitykselle ja konkreettiselle toteutustapojen testaamiselle (Kuva 4).



Kuva 4. Mandrivan työpöytä KDE:llä

4 KEHYSTEKNIIKOIDEN VALINTA JA KÄYTTÖÖNOTTO

Kehystekniikoilla tarkoitan sovelluksia, komponentteja ja asetuksia, jotka olivat välttämättömiä kokonaisuuden kannalta. Loppukäytössä kaikkien osien toimintaa ei edes välttämättä huomaa, jollei niistä tiedä. Joidenkin osien valinta oli melko helppoa, toisten taas vaati jonkun verran kokeilua.

4.1 Rakenteiden alkusuunnittelu

Toimivan kokonaisuuden edellytyksenä on edetä osa kerrallaan eteenpäin. Valintojen motiiveina toimivat useat tekijät, kuten niiden käyttökelpoisuus ja oma osaaminen niiden saralla. Kaikkien valintojen lähtökohtina ovat kuitenkin jotkin oleelliset asiat, jotka tekevät ne tarpeellisiksi.

4.1.1 Sivustojen tyypin valinta

Ensimmäinen vaihe toimivan sivuston luomisessa oli sisällön tyypin valinta. Varsinaisina harkittavina vaihtoehtoina olivat PHP, Flash ja JavaServer Pages (JSP). Ennen kuin valitsee jonkun aiemmin mainituista, on hyvä perehtyä niiden ominaisuuksiin, sekä hyviin ja huonoihin puoliin.

PHP (PHP: hypertext processor) on hyvin laajasti web-sivustoilla käytetty ohjelmointikieli. PHP:n ideana on laaja luokkakirjasto, joka sisältää hyvin monenlaisia funktioita. PHP:n hyvinä puolina pidetään sen helppoa lähestyttävyyttä, sen sisältämien funktioiden määrää ja sitä, ettei se vaadi asiakasohjelmalta muuta kuin tavallisen selaimen. Huonoina puolina taas pidetään nimiavaruuksien puutetta ja tietokantarajapinnan epäselvyyttä. Lisäksi sivu latautuu kokonaan vasta kun PHP on suorittanut toimintansa loppuun asti ja toiminnot suoritetaan vain latauksen yhteydessä. /20/

Flash on tapa julkaista tietoa, pääosin multimediaa. Sisältö koostetaan esimerkiksi Macromedia Flash:lla (nykyisin Adobe Flash) ja julkaistaan tämän jälkeen sivuille. Hyviksi puoliksi Flashissa katsoimme sen tietoturvan, sillä julkaistuja swf-

tiedostoja ei voi jälkeenpäin muuttaa. Se sisältää multimedian kannalta hyviä ominaisuuksia, joita ovat valmiit komponentit esimerkiksi soittimiin. Kaiken voi lisäksi luoda graafisella työkalulla, jota on melko helppo käyttää. Huonoina puolina Flashissa voi pitää sen tarvetta erilliselle playerille asiakassovellukseen. Flashien luontiin tarvittava ohjelma on myös hyvin kallis yksityiseen käyttöön. Ehkä huonoin piirre Flasheissa on se, että ne suoritetaan asiakkaan laitteella, jolloin saattaa tulla ongelmia mobiililaitteilla. /21/

JSP vastaa tyyliltään hyvin paljon PHP:tä, koska myös sen olemukseen kuuluu luoda dynaamisia sivuja. JSP:tä luodaan lisäämällä JSP:ksi merkattua sisältöä html:n joukkoon. Kieli on hyvin monipuolinen ja sillä voi luoda helposti sivuille dynaamisuutta. Se sisältää hyvin paljon erilaisia funktioita moneen tarkoitukseen. Kielen käyttö on helppoa sellaiselle, joka on jo aiemmin ohjelmoinut jollain Javaan pohjautuvalla kielellä. JSP-sivut vaativat kuitenkin toimiakseen sovelluspalvelimen kuten Apache Tomcat:n. Palvelimen vaihtamisella Tomcattiin olisi vaikutuksia myös resurssien käyttöön, sillä se on käytössä paljon raskaampi kuin tavallinen http-palvelin. Sivuja voi myös olla vaikea tehdä JSP:llä, jos aiempaa kokemusta Javasta ei ole. /22/

4.1.2 Johtopäätökset erilaisista kielistä

Kuten aiemmasta osiosta selviää, jokaisella ohjelmointitavalla on omat hyvät ja huonot puolensa. Tämän tiedon valossa onkin todettava, että parhaaseen lopputulokseen pääsee luultavasti yhdistelemällä joitakin niistä sellaisiin kohtiin, joissa niiden ominaisuudet ovat parhaimmillaan ja oma taitotaso riittää.

4.1.3 Ratkaisu rakenteen suhteen

Lopullinen päätös oli käyttää PHP:tä pääasiallisena kielenä sivustojen luontiin. Flashia on hyvä käyttää joidenkin ei-pakollisten multimediakomponenttien esit-

tämiseen ja käyttämiseen. Paras tulos PHP:n kanssa toimittaessa saadaan käyttämällä rinnalla MySQL-tietokantapalvelinta tai muuta tietokantaa. Kyseinen yhdistelmä tunnetaan myös nimellä LAMP (Linux, Apache, MySQL, PHP).

4.2 PHP:n ja MySQL:n asennus

Helpoin tapa Mandrivassa asennella eri ohjelmistoja on käyttää urpm:ää. Tätä varten ennen käyttöä on hyvä asettaa mirrorit kohdalleen. Ensiksi kannattaa poistaa vanhat urpm-lähteet komennolla *urpmi.removemedia – a* ja tämän jälkeen käydä osoitteessa http://easyurpmi.zarb.org/ hakemassa itselleen sopivat mirrorit. Lopuksi liitetään aiemmin mainitulta sivulta saatu tekstipätkä linux:n konsoliin ja urpm-lähteet ovat kunnossa.

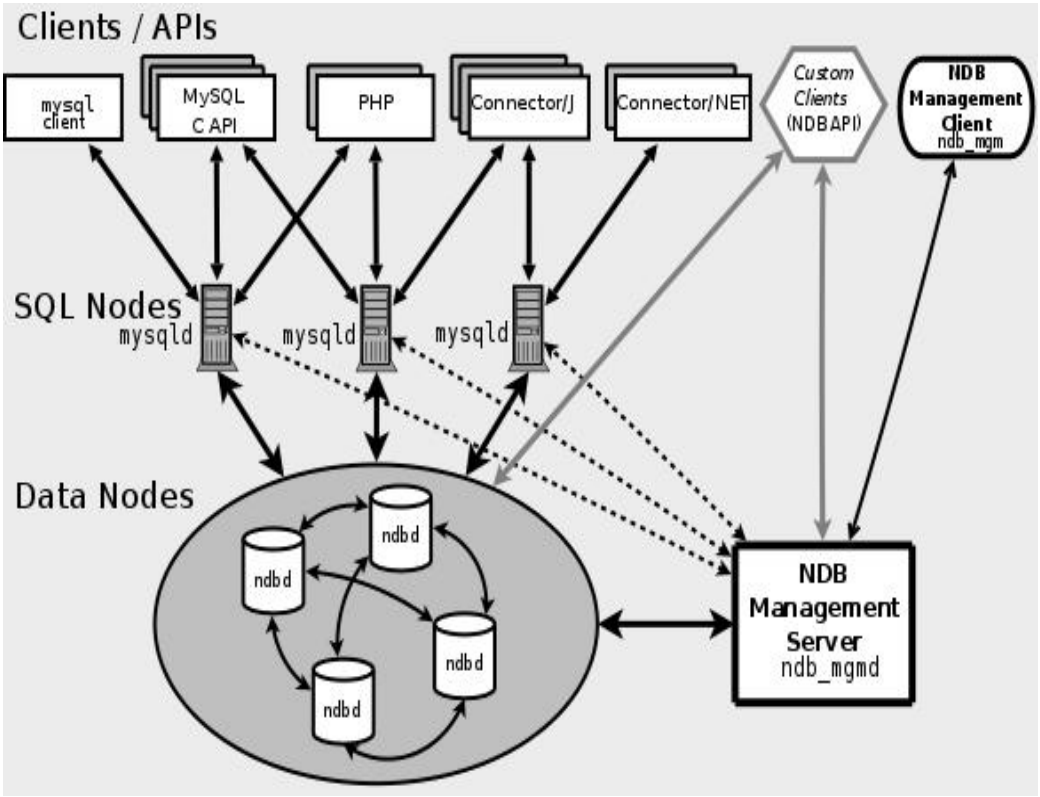
PHP:n asennus on hyvin yksinkertaista, sillä asennus onnistuu suoraan antamalla komennon: *urpmi php5*. Tämä saa aikaan sen, että urpm-kannasta haetaan vastaavuutta kohteelle php5. Näitä ovat esimerkiksi: php5, libapache2-mod-php5 ja php5-common. Kyseiset paketit koskevat siis Apachen versiota 2 ja php:n versiota 5. Asennuksen jälkeen tulee käynnistää web-palvelin uudelleen. Toiminnan voi todeta tekemällä esimerkiksi testisivun, johon lisää seuraavanlaisen kohdan: *<?php phpinfo(); ?>*. Jos php toimii moitteettomasti, tulevat tiedot php:stä ja siihen liittyvistä moduuleista.

MySQL:n asennus tapahtuu suurin piirtein samalla tavalla kuten php:nkin. Kuva 5 on otettu Mandriva 2007 free:stä ja asennettavana oli MySQL5.

```
[root@dsl-tregw3-fe59f800-11 ville]# urpmi mysql
Yksi seuraavista paketeista tarvitaan:
 1- MySQL-5.0.24a-2mdv2007.0.i586 : MySQL: a very fast and reliable SQL database
 engine (to install)
 2- MySQL-Max-5.0.24a-2mdv2007.0.i586 : MySQL - server with extended functionali
ty (to install)
Mikä on valintasi? (1-2) 1
To satisfy dependencies, the following packages are going to be installed:
MySQL-5.0.24a-2mdv2007.0.i586
MySQL-client-5.0.24a-2mdv2007.0.i586
MySQL-common-5.0.24a-2mdv2007.0.i586
libmysql15-5.0.24a-2mdv2007.0.i586
perl-DBD-mysql-3.0006-2mdv2007.0.i586
perl-DBI-1.52-1mdv2007.0.i586
Proceed with the installation of the 6 packages? (40 MB) (K/e)
```

Kuva 5. MySQL:n asennus

Kuvaan 5 nähden erona oli myös yksi vaihtoehto lisää, joka oli NDB-versio. NDB:n etuna ovat tietokannan ulkoiset haut. Kuva 6 selventää NDB:n toimintaa.



Kuva 6. NDB:n toiminta(http://little-cow.sianscripts.com/?q=node/14)

Kuten kuva 6 havainnollistaa, NDB mahdollistaa tietokannan ulkoiset haut ja sen ylläpidon siirron sekä mahdollisen siirron vaikuttamatta toiminnassa olevaan tietokantaan. Varmuutta tämäntyyppisen tietokannan tarpeesta ei ollut, mutta se lisäsi mahdollisuuksia. Asennus etenee omalla painollaan tämän jälkeen. Lopuksi tulevat vielä tärkeät tiedot kyseisestä asennuksesta. /11/

```
Lisätietoja paketista MySQL-5.0.24a-2mdv2007.0.i586

The initscript used to start mysql has been reverted to use the one shipped
by MySQL AB. This means the following changes:

 * The MYSQLD_OPTIONS="--skip-networking" option in the /etc/sysconfig/mysqld
   file has been removed, this is now set in the /etc/my.cnf file.

 * The MySQL Instance Manager is used by default, set use_mysqld_safe="1" in
   the /etc/sysconfig/mysqld file to use the old mysqld_safe script.

The extra MySQL-NDB server package has been merged into the MySQL-Max package
and ndb related pieces has been split into different sub packages as done by
MySQL AB. The MySQL libraries and the MySQL-common sub package uses the
MySQL-Max build so that no functionality required by for example the NDB
parts are lost.

The MySQL-common package now ships with a default /etc/my.cnf file that is
based on the my-medium.cnf file that comes with the source code. The
/etc/my.cnf  file is constructed at build time of this package.

To connect to the Instance Manager you need to pass the correct command line
options like in the following examples:

  * mysql -u root --password=my_password --port=2273 --protocol=TCP
  * mysql -u root --password=my_password --socket=/var/lib/mysql/mysqlmanager.so
ck

Please note you also need to add a user in the /etc/mysqlmanager.passwd file and

make sure the file is owned by the user under which the Instance Manager service

is running under.
```

Kuva 7. MySQL:n asennuksen lopputeksti

Lopuksi tulevaa tekstiä (kuva 7) on hyvä tutkia, sillä välillä tulee muutoksia, joita ei aina osaa odottaa. Esimerkkinä 4-versiosta siirryttäessä 5:een skip_networking parametri on eri tiedostossa, jolloin tietokannan etähallinta on mahdotonta. Tällöin MySQL ei kuuntele ollenkaan porttia 3306 (oletus-portti). /11/

Yksi paketti, joka täytyy asentaa toiminnan varmistamiseksi, on *php-mysql*. Kyseinen on connectori PHP:n ja MySQL:n välille. Se mahdollistaa näiden kahden toiminnan keskenään. Connectori eli ”yhdistin” on eräänlainen tulkki näiden kahden sovelluksen välille, joka kykenee välittämään tietoa ja käskyjä kahteen suuntaan.

4.3 PHP:n ja MySQL:n asetusten määrittely

PHP:n asetuksia muutetaan *etc/php.ini*-tiedostoa muuttelemalla. Tiedosto on tekstipohjainen ja hyvin kommentoitu, joten sen käsittely oli melko helppoa. Yksi käyttömukavuutta helpottava rivi on: *short_open_tag = On*. Tämä mahdollistaa lyhyiden php-tagien käytön. Käytännössä php-osion merkkaamiseen voi siis käyttää *<? php-koodia ?>*, eikä tarvitse käyttää muotoa *<?php php-koodia?>*. On myös hyvä ottaa huomioon sellaiset käyttäjät, joilla ei ole y2k-yhteensopivia selaimia (nykyään ne ovat luultavasti melko vähissä). Tämä toteutetaan rivillä *y2k_compliance = On*. Sivustojen testaamista helpotti se, että näki syntyneet virheet. Virheet ja varoitukset sai päälle ensiksi valitsemalla kaikki virheet ja varoitukset raportoitavaksi, *error_reporting = E_ALL* ja tämän jälkeen vielä asettamalla virheiden esityksen sivustossa, *display_errors = On*. Kuitenkin kannattaa ottaa nämä varoitukset pois sivujen valmistumisen ja testaamisen jälkeen, sillä ilmoitukset saattavat sisältää tärkeitä tietoja palvelimesta, jotka ei ole tarkoitettu ulkopuolisille. Loput asetuksista on hyvä muuttaa tarpeen tullen sivuilla käyttäen ini_set-funktiota.

MySQL:n asetusten tarkistamisessa tärkeintä on tarkistaa, ettei skip_networking-komento ole käytössä asetus-tiedostossa. MySQL:n versiota 4 käytettäessä tieto sijaitsee */etc/sysconfig/mysqld*. Versiota 5 käytettäessä tieto löytyy tiedostosta */etc/my.cnf*. Tärkeää on myös määrittää tietokannan root-käyttäjälle salasana. Se tapahtuu käyttämällä mysqladmin ohjelmaa. Täsmällisesti ilmaistuna komento on *mysqladmin –u root –password='salasana'*. Tämän jälkeen kannattaa luoda tavallinen käyttäjä, jolla on joitakin oikeuksia. Esimerkkinä on käyttäjän luonti jolla on kaikki oikeudet:

1. Siirrytään mysql hallinnointiin: *mysql –u root –p*. Hyväksymällä aiemman järjestelmä kysyy salasanaa.

2. Määritellään käyttäjä ja tämän oikeudet, esimerkiksi käyttäjä jolla on kaikki oikeudet: *GRANT ALL PRIVILEGES ON *.* TO 'käyttäjä'@'localhost' IDENTIFIED BY 'salasana' WITH GRANT OPTION;*
3. Lopetetaan ohjelma komennolla *quit;*

Jos käyttäjän halutaan käyttävän tietokantaa ulkoisesti, täytyy edetä aiempien kohtien mukaan, mutta kirjoittaa vaihe 2 uudestaan sillä erotuksella, että 'käyttäjä'@'localhost' – kohtaan tulee localhostin tilalle ulkoinen kohde tai villikorttimaski, joka on %. Alussa määritellään *ALL PRIVILEGES*, joka määrittää kaikki toiminnot käyttäjälle. Kyseiseen kohtaan voi kuitenkin halutessaan määritellä sellaisia käskyjä kuten SELECT, DROP ja DELETE. Käyttäjälle voisi myös määrittää mihin, tauluihin sillä on oikeuksia. Tämä on merkitty aiemmassa mallisa *.*:llä. Esimerkiksi, jos halutaan antaa käyttäjälle oikeus mysql-tietokannan user-tauluun, niin se tulee muodossa mysql.user.

4.4 PHP:n, MySQL:n ja apachen toiminnan testaus

Lopuksi oli vielä tarve testata kaikkien kolmen komponentin yhteistoiminta. Tätä varten oli luotava testitietokanta, jonne oli sijoitettava jokin arvo. Tämän jälkeen tarvitsi enää tehdä sivu, jossa kyseinen tieto haettaisiin taulusta. Loin seuraavanlaisen taulun:

```
CREATE TABLE `testi`.`testikanta` (
  `sarake1` VARCHAR(45) NOT NULL,
  `sarake2` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`sarake1`)
)
ENGINE = InnoDB;
```

Esimerkki 1. Taulun luonti

Kyseinen taulu (esimerkki 1) sisältää kaksi saraketta, jotka molemmat ovat muuttuvan mittaisia merkkijonoja./11/

Sivulle tulee lisätä PHP:n MySQL:n yhdistävä lause ja tietokannan valinta:

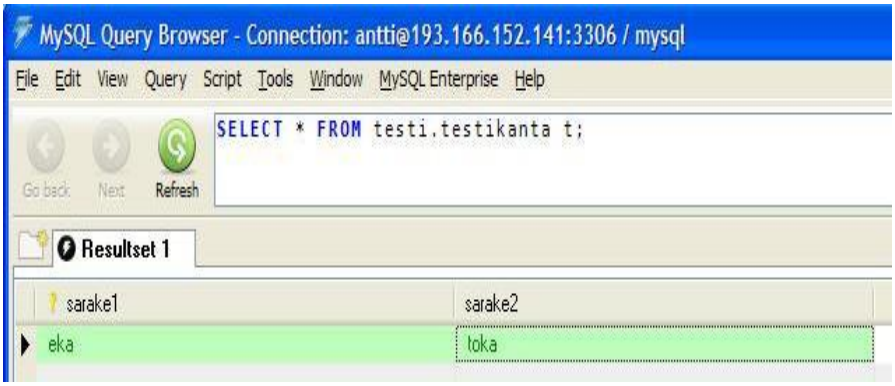
```
$conn = mysql_connect('193.166.152.141', 'antti', 'tekpo03') or die
('Virhe yritättäessä yhdistää');
        mysql_select_db('muser');
```

Tämän jälkeen voidaan suorittaa kysely

```
$kysely="SELECT sarake2 FROM testikanta WHERE sarake1 =
'eka'";
$vastaus = mysql_query($kysely, $conn);
$palautus = mysql_fetch_row($vastaus);
echo $palautus[0];
```

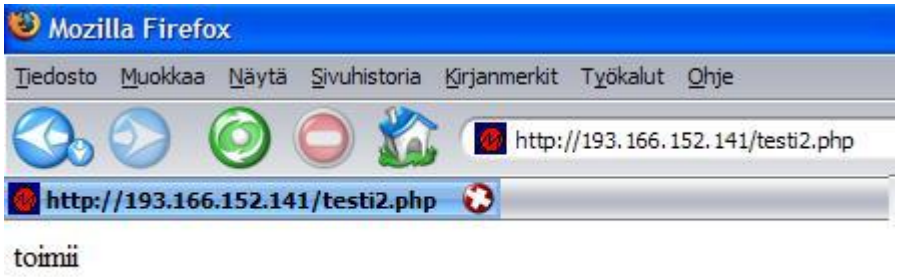
Esimerkki 2. Kyselyn suorittaminen PHP:llä /18/

Esimerkillä 2 saadaan palautuvasta tulosjoukosta ensimmäinen alkio. Taulun sisältö on esitetty kuvassa 8.



Kuva 8. Testikanta-taulun sisältö

Tarkoituksena on saada kuvan 8 pienellä testillä aikaan kuvan 9 mukainen tuloste.

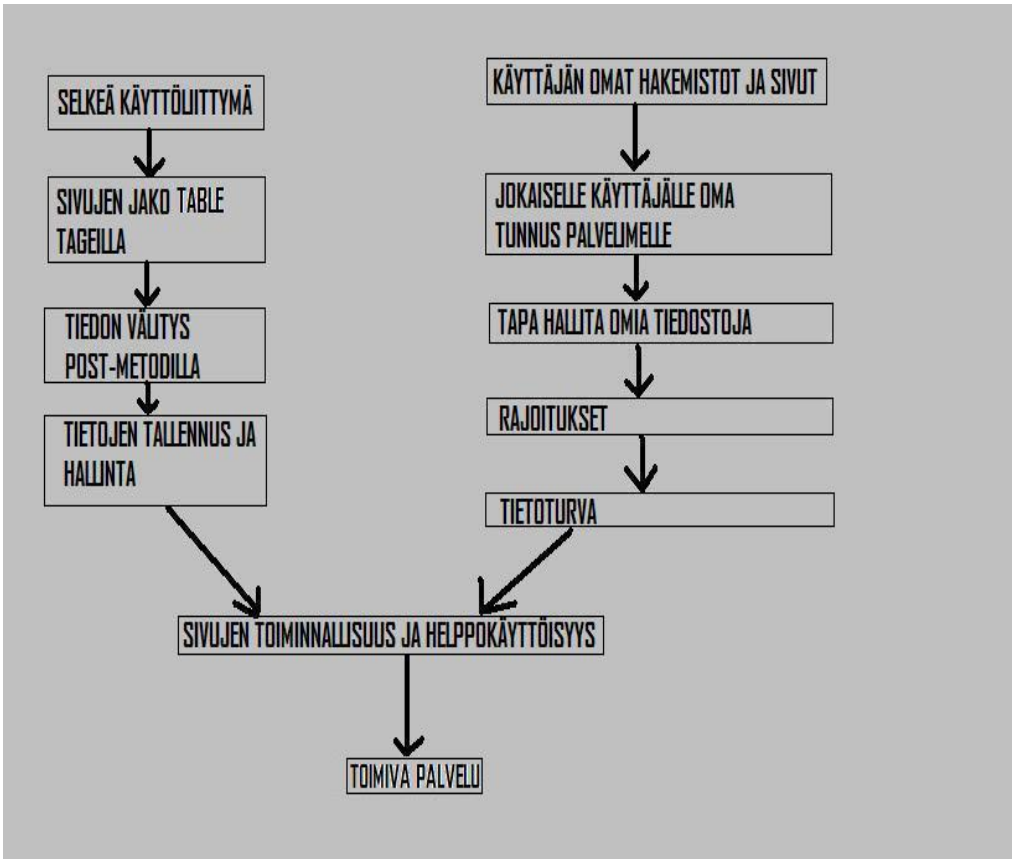


Kuva 9. Testin tulos

Lopputuloksena voidaan todeta, että asennetut osat toimivat halutulla tavalla.

4.5 Sivujen suunnittelu

Suunnittelu aloitettiin miettimällä haluttua lopputulosta. Lopputuloksen tulisi olla suhteellisen helppokäyttöinen, helposti muunneltava ja ennen kaikkea riippumaton vastapäästä, joka tässä tapauksessa tarkoittaa matkapuhelinta. Tiedostimme kuitenkin sen seikan, että emme voisi luoda yhteensopivuutta aivan jokaisen puhelinmallin kanssa. Vastakkain olivat kaksi eri puolta: sivujen selailu puhelimella ja palvelimen käyttö pelkkänä tietovarastona. Kyseisessä asiassa päädyimme kompromissiin siinä suhteessa, että päätimme luoda käyttäjälle karsitun version sivuista puhelimella selailtavaksi, mutta joka kuitenkin vaatisi xhtml-tasoisen selaimen. Luonnostelimme yllä olevista ajatuksia kuvan 10.



Kuva 10. Sivujen toimintakaavio

Kuva 10 esittää sivustojen luonnin kahta eri puolta: vasemmalla toiminnallisuus ja oikealla helppokäyttöisyys. Toiminnallisuus tarkoittaa fyysistä toiminnallisuutta, tarkemmin ottaen toimintavarmuutta ja siihen liittyvää sujuvuutta. Helppokäyttöisyys taas pitää sisällään asioita, jotka kuuluvat yleisiin kompastuskiviin käyttömukavuudessa. Nämä kompastuskivet eivät tarkoita, että niihin kuuluvia asioita ei tulisi olla sivuilla. Päinvastoin, ne ovat juuri keskeisessä osassa, mutta niihin tulee kiinnittää erityistä huomiota toteutettaessa. Kaikista vaikeimpana osuutena ennen toteutusta pidettiin tietoturvan osa-aluetta. Siinä käyttömukavuus ja tietoturva vääntävät kättä. Kuten Juha Niemi sanoi: ”käyttömukavuus kertaa tietoturva on vakio.” Se tarkoittaa, että toisen vähentäminen lisää toista ja päinvastoin.

Hyvin merkittävä kysymys toteutuksen suhteen oli käyttäjän pääsy omiin tiedostoihinsa. Tässä tilanteessa päätimme alustavasti turvautua suhteellisen yksinker-

taiseen ratkaisuun, käyttämään apcaheen luotua modia käyttäjähakemistoja (apache-mod_userdir).

4.5.1 Käyttäjähakemistojen käyttö

Käyttäjähakemistojen käyttö on hyvin yksinkertaista, sillä lisäosan käyttöönoton (urpmi – asennus) jälkeen jokaisen */home/* polun takana olevalle käyttäjälle voidaan luoda hakemisto, johon pääsee verkon kautta http-protokollan avulla. Kyseinen hakemisto on vakiomuotoinen, eikä sitä tarvitse erikseen määritellä käyttöön. Hakemiston voi nimetä joko *public_html* – nimellä tai lyhyemmin *pub*. Polku on kokonaisuudessaan tämän jälkeen */home/käyttäjä/public_html*. Http:n avulla hakemistoon pääsee käyttämällä käyttäjänimen edessä mato-merkkiä(~). Polku on siis kokonaisuudessaan *http://palvelin/~nimi/*. Tämän tyylin käytössä kuitenkin piilee ilmeinen tietoturva-ongelma. Kaikki tiedostot public_html-hakemistossa ovat kaikkien saatavilla, ei muokattavissa, mutta saatavissa. /3/ Koimme kuitenkin pulmalliseksi toteuttaa suhteellisen helposti sellaista suojausta, jossa tiedostot eivät olisi yleisesti saatavilla, mutta kuitenkin matkapuhelimelle mahdollisia. Kohtuullista ratkaisua ei löytynyt helposti, joten päätimme jättää tietoturvan siltä osilta ainakin vähäksi aikaa.

Edut hakemistojen käytöstä olivat kuitenkin huomattavat. Kaikista suurin etu oli se, että käyttäjän musiikin pystyi jaottelemaan omien hakemistojensa perusteella, eikä näin ollen ollut tarvetta jatkuvasti pitää tietoa kappaleista ja niiden omistajista. Toinen lähes yhtä suuri etu oli mahdollisuus käyttää SSH-ohjelmia tiedostojen siirrossa oikeisiin paikkoihin. Jos kaikilla olisi ollut yhteinen paikka tiedostoille, olisi ongelmaksi muodostunut niiden lataamisen estäminen muilta käyttäjiltä.

4.5.2 Skriptien käyttö

PHP:llä pystyy käyttämään melko monia järjestelmän käskyjä funktioiden avulla. Kaikkia ei siis silläkään voi tehdä, joten täytyy luoda järjestelmään omia skriptejä. Skripti vastaa käyttöjärjestelmään suoraan kirjoitettuja komentoja. Skripti voi kuitenkin sisältää monta peräkkäistä käskyä, jotka suoritetaan rivi riviltä, ylhäältä alas. Skripti voi myös sisältää ehtolauseita ja silmukoita, kuten *if, for* ja *while*. Hyödyllisen skriptistä tekee se, että se voi käyttää komentoriviargumentteja. /15/

Esimerkki skriptin toimintaa voi testata tekemällä yksinkertaisen skriptin, joka kaiuttaa syötetyn argumentin. Aluksi on hyvä määritellä terminaalin tyyppi. Huomasimme käyttäessämme PHP:stä käsin skriptejä, että jos terminaali-tyypin jätti määrittelemättä, niin skripti ei välttämättä toiminutkaan. Määrittely tapahtuu laittamalla alkuun *#!terminaalin-sijainti* eli *#!/bin/sh*. Seuraavaksi voidaan kaiuttaa annettu parametri. Argumentteihin viitataan *$numero*, jossa numero tarkoittaa argumentin paikkaa. Numeron ollessa 0 vastaa se itse skriptiä. Numero 1 on ensimmäinen argumentti. Esimerkissä 3 on luotu testiskripti ja se on suoritettu antamalla argumentti *testilause*.

```
[root@mstream bin]# cat testiskripti
#!/bin/sh
echo $1
[root@mstream bin]# ./testiskripti testilause
testilause
```

Esimerkki 3. Skripti ja sen suoritus

Käyttökelpoisia erikoismerkkejä komentoriviargumenttien käsittelyyn on lisäksi *$#* ja *$@*. $# tarkoittaa argumenttien määrää, $@ taas tarkoittaa kaikkia annettuja argumentteja. /16, 17/

Muita tärkeitä kohtia Linuxin skriptejä tehtäessä on huomata esimerkiksi lainausmerkkien erot. Tuplalainausmerkit (”) tarkoittavat, että sen sisällä on merkkijono. On syytä kuitenkin tietää, että $- ja \-merkki toimivat silti kuten normaalisti (muuttujana tai erikoismerkin etuosana). Yksittäinen lainausmerkki (’) pitää kai-

ken sisällään olevan muuttumattomana. Taaksepäin kallellaan olevat lainausmerkit (`) suorittavat komennon. Esimerkiksi laskutoimitus: *tulos = `expr $luku1 + $luku2`*. /17/

4.5.3 PHP:n käyttö

PHP:n vahvuus ovat sen lukuisat valmiit funktiot. Niiden ansiosta monet muuten vaikeat asiat, kuten merkkijonojen erikoiskäsittely tai niiden suhteellisen vertailun tekeminen vaatisi kehittäjältä omaa aikaansa.

Aivan kuten Linuxin skriptien teossa myös PHP:ssä muuttujat merkataan $-merkillä. Muuttujia ei myöskään tarvitse erikseen alustaa, vaan ne alustuvat käytettäessä. Jokainen rivi päättyy puolipisteeseen (;). Aiemmin mainitsin tavasta merkitä php:tä tageilla (<? php:tä ?>), kun PHP-alueen sisään kirjoittaa html-tageja, joissa on attribuutteja, tulee olla käyttämättä lainausmerkkejä, sillä niillä merkitään ei-PHP tietoa (html). Esimerkiksi jos merkitsee lomaketta: *echo "<form method=POST> … </form>";* on oikeia tapa. Peräkkäisiä muuttujia merkitessä tulee huomata erottaa ne pisteellä: *echo $muuttuja1 . $muuttuja2*;. /19/

4.5.4 PHP:n ja Linux skriptien yhteiskäyttö

Yhteiskäytöllä saavutetaan huomattava etu verrattuna yksipuoliseen käyttöön. Linux skriptien käyttö tarkoittaisi, että käyttäjän tulisi olla kirjautuneena palvelimelle esimerkiksi SSH:n avulla ja suorittaa skriptejä. Näin ollen käyttäjällä olisi osittainen suora pääsy tiedostoihin ja myös mahdollisuus tutkia tietoturva-aukkoja. Pelkkä PHP:n käyttö hankaloittaisi joidenkin toteutusten tekoa, kuten yksittäisen käyttäjän omaa aluetta.

Skriptien käyttö PHP:n avulla onnistuu käyttämällä funktiota *exec*. Kyseinen funktio mahdollistaa käyttöjärjestelmän komentojen ja skriptien käytön. Samassa tilanteessa on myös hyvä käyttää *sudo*:a skriptien suoritukseen. Sudo on komento, joka mahdollistaa tavallisen käyttäjän käyttää pääkäyttäjän komentoja. Sudo:n käyttäjät ja tiedot komennoista, joita on mahdollista käyttää, tallennetaan */etc/sudoers*-tiedostoon. Esimerkissä 4 on meidän palvelimellemme muodostettu *sudoers*-tiedosto.

```
[root@mstream antti]# cat /etc/sudoers
# sudoers file.
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the sudoers man page for the details on how to write a sudoers file.
#

# Host alias specification

# User alias specification

# Cmnd alias specification

Cmnd_Alias SU = /bin/su

# Defaults specification
# Defaults syslog=auth

# Runas alias specification

# User privilege specification



# Uncomment to allow people in group wheel to run all commands
# %wheel        ALL=(ALL)       ALL

# Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL

# Samples
# %users  ALL=/sbin/mount /cdrom,/sbin/umount /cdrom
# %users  localhost=/sbin/shutdown -h now
%apache ALL=/bin/recho, /bin/rm, /bin/cat, /bin/scripti1, /bin/chpasswd, /bin/scripti2,
/usr/sbin/chpasswd, /bin/cp, /bin/movemp3, /bin/mv, /bin/tuhomp3, NOPASSWD: ALL
```

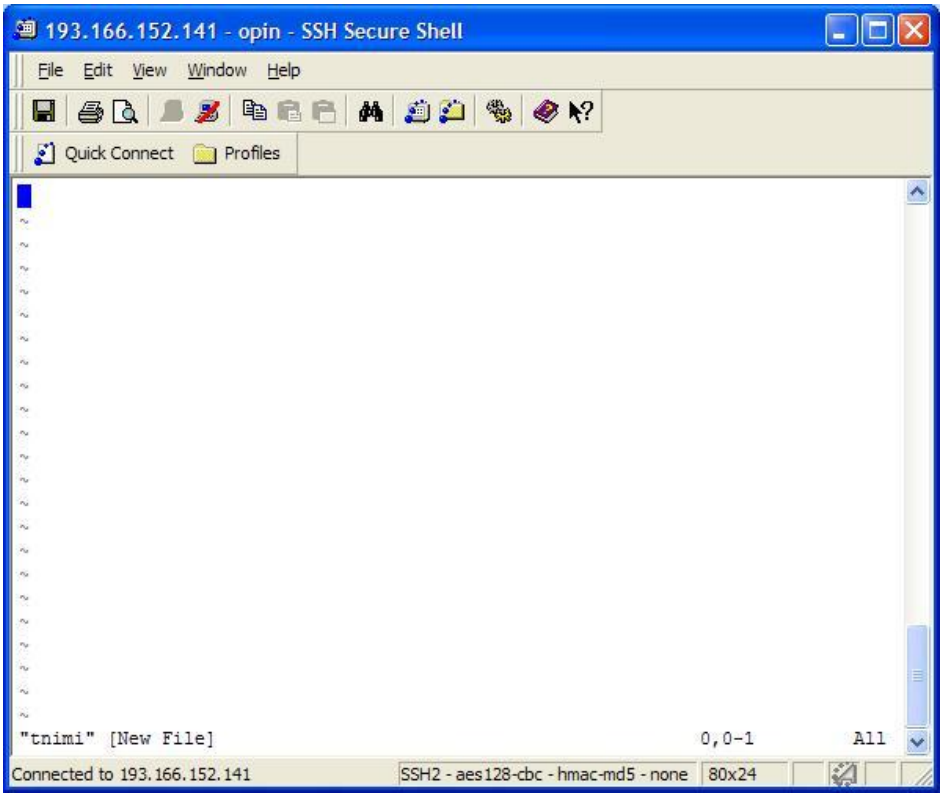
Esimerkki 4. Sudoers-tiedoston sisältö

Tärkeintä sudoers-tiedostoa luotaessa on muokata sitä visudo nimisellä ohjelmalla. Se on vi-tekstieditori, joka avaa automaattisesti sudoers tiedoston. /6, 7/ Sen käytöstä selostan lisää hieman myöhemmin.

Aluksi emme olleet varmoja, oliko käyttäjänä apache, kun skriptiä yritettiin suorittaa web-sivulta. Niinpä kokeilimme testiskriptin avulla ja tutkimme *messages*-tiedostoa. Tiedon sieltä sai kaivettua käyttämällä *grep*-komentoa, joka etsii tiedostosta annettua sanaa vastaavaa merkkijonoa. /6, 7/ Komento oli: *grep sudo /var/log/messages*. Sillä ilmeni, että käyttäjä web-sivulta käytettynä todellakin oli apache.

Meidän tuli siis seuraavaksi sallia apache-käyttäjä sudoers-tiedostossa. Päätimme kuitenkin sallia apache-ryhmälle oikeudet. Kyseisellä järjestelyllä pyrimme hakemaan muokattavuutta tarpeen mukaan. Toisin sanoen halusimme jättää oven auki muutoksille. Sudoers-tiedostossa käyttäjän ja ryhmän ero on % -merkki ryhmän edessä. Määrittely tapahtuu seuraavanlaisesti: käyttäjä/ryhmä sallittu, sallittu, NOPASSWD: ALL. Viimeinen osa tarkoittaa, että ei tarvitse välittää salasanaa komentoja käytettäessä. /6, 7/

Vi-editori ei ole ehkä kaikista helpoin tekstieditori. Sen etuna on kuitenkin, ettei se luo tiedostoon ylimääräisiä merkkejä tai rivinvaihtoja. Editori käynnistetään komennolla: *vi tnimi*.



Kuva 11. Vi-editori

Kuvassa 11 näkyvät ~ -merkit tarkoittavat, ettei riviä ole olemassa. Tekstin syöttämiseksi on painettava i-kirjainta. Sen jälkeen tekstiä voi syöttää normaalisti. Lopetettaessa tekstin syöttötilasta pääsee pois Esc-näppäimellä. Komento syötetään painamalla : -merkkiä. Halutessaan poistua tallentamatta täytyy laittaa *q!*, tallennettaessa *wq*. /7/

4.6 Arviointi kehysrakenteiden toteutuksesta

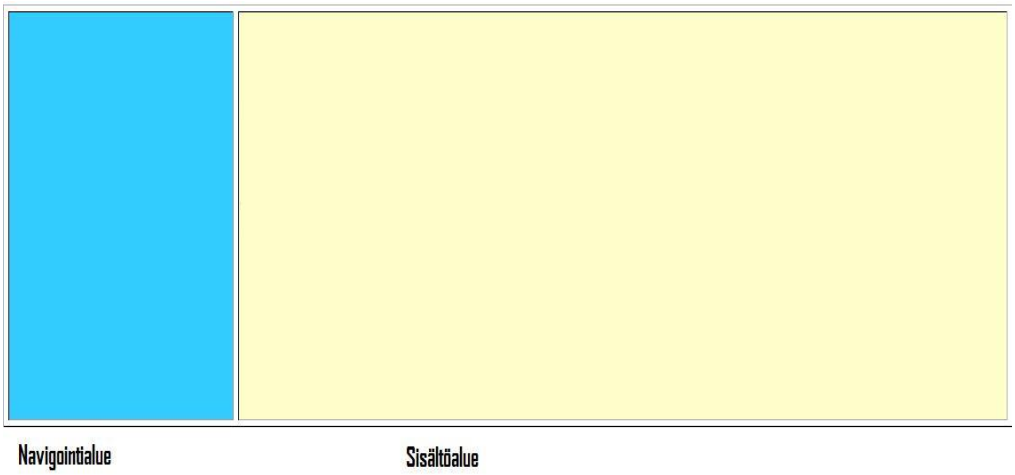
Kovin suuria tai ylitsepääsemättömiä ongelmia ei kohdattu toteutettaessa. Kaikkien osien toteuttaminen vaati niihin perehtymistä ja pieniä kokeiluja, joilla sai hyvin selville niiden luonteen. Jälkikäteen arvioituna yksikään valinnoista ei ollut huono tai sellainen ettei sillä olisi ollut tarvittavia ominaisuuksia. Olimme siis valmiita siirtymään sisällön tuottamiseen.

5 VERKKOPALVELUN LUONTI

Käytän sivustosta nimeä palvelu, sillä sen tarkoituksena oli kyetä luomaan jotain mitä ei aiemmin ollut olemassa. Tehtävänä oli aluksi suunnitella ulkoasu, joka takaisi toiminnan ja olisi käytännöllinen. Seuraavana vuorossa oli käyttäjätunnuksien luontipalvelu. Viimeisenä suunniteltaisiin käyttäjille omat sivut, joissa olisi musiikinhallintaan tarvittavia toimintoja. Näitä toimintoja ovat esimerkiksi soittolistojen luonti ja niiden muokkaaminen. Kehitystyökaluna käytimme Macromedia Dreamweaveria, koska sillä sivujen luominen käy hieman nopeammin ja sen lisäksi siitä oli aiempaa kokemusta.

5.1 Ulkoasu

Joskus hyvistäkin verkkosivuista tulee huonoja niiden kehnon ulkoasusuunnittelun takia. Ne ovat joko liian täynnä informaatiota tai se on liian vaikeasti löydettävissä. Päätimme jakaa sivuston kahteen palstaan. Vasemman puoleinen palsta toimisi eräänlaisena navigointi alueena ja oikeanpuoleinen tiedon esittämisen alueena. *Table*-tageja käytettiin sivun rakenteeseen *frame*-tagien sijaan. Syynä siihen oli pieni ero lopputuloksessa ja frame:ien vanhakantaisuus. Ulkoasu on esitetty kuvassa 12.



Kuva 12. Ulkoasun pohja

Värien valinta saattaa tuntua pieneltä asialta, mutta huonosti valitut värit haittaavat sivujen käyttöä. Yleisesti ottaen hillityt värit ovat selkeimpiä ja rasittavat vähiten silmiä. Valitsimme sivuillemme väreiksi turkoosin ja beigen. Turkoosia käytettiin navigointialueen pohjavärinä ja beigeä sisältöltöalueen. Myös värien keskinäisellä kontrastilla on merkityksensä selkeyden kannalta, sillä se rajaa hyvin alueet. Rauhallisilla väreillä oli myös merkitystä itsellemme, sillä tuskin olisimme jaksaneet testailla sivuja pitkään, jos värit olisivat olleet räikeitä.

5.2 Käyttäjätunnusten luonti

Tunnusten luonti haluttiin toteuttaa suoraan sivustojen kautta. Suunnitelmamme oli, että käyttäjän tiedot tallentuisivat suoraan tietokantaan ja kuitenkin samalla tulisi myös eräänlainen rajattu käyttäjä palvelimelle.

Aluksi luomamme tunnuksentekopalvelu (kts. Liite1) oli avoin, mutta nopeasti havaitsimme sen haitallisuuden palvelimelle. Vaikka sivuillemme pääsi vain tietämällä osoitteen, huomasimme, että joitakin hakkereita oli jo yrittänyt murtautua palvelimelle suoraan. Niinpä päätimme, että tunnuksen luonti vaatisi erillisen salasanan.

5.2.1 Tunnuksen luonti Linuxiin

Tavallisin tyyli luoda tunnus on käyttää komentoa *adduser*, johon voi lisätä monia parametrejä. Tätä ennen kuitenkin loimme tavallisille käyttäjille yhteisen ryhmän nimeltään *muser*. Sen luominen tapahtui komennolla: *groupadd –g 501 muser*. Jokaisella ryhmällä on oltava oma uniikki tunnus, joka aiemmassa esimerkissä luotiin parametrillä *–g* ja arvoksi annettiin 501.

Halusimme pelkän tunnuksen lisäksi luoda käyttäjälle omat hakemistot ja terminaaliksi pelkän sftp:n. Tällöin komento saisi muodon: *adduser –g 501 –m –s /usr/lib/ssh/sftp-server tunnus*. /6/

Salasanaa käyttäjälle ei voi antaa suoraan aiemmalla komennolla, sillä sen tulee olla kryptattu. Tämä täytyi siis toteuttaa toisella tavalla. Ehkä paras tapa saada syötettyä selväkielinen salasana oli käyttää *chpasswd*-komentoa. Sen erikoisuutena on, että salasana ja sitä vastaava tunnus voidaan lukea tiedostosta. Salasanan ja tunnuksen on oltava muodossa *tunnus:salasana*. Tiedostosta luetaan salasana chpasswd-ohjelmaan. Tämän tapahtuu seuraavanlaisesti: *cat tiedosto | chpasswd*. /6, 7/

Viimeisenä tulee vielä luoda käyttäjälle tarvittavat hakemistot myöhempää käyttöä varten. Näitä kansioita ovat *public_html, mp3* ja *listat*. *public_html*. Kansiota tarvitaan, jotta selaimella pystyy erottelemaan helpommin käyttäjät. Kansioon kopioidaan uusi sivu, joka sisältää käyttäjän tarvitsemat toiminnot. Muiden kansioiden tarkoituksesta kerron myöhemmin. Mainituista komennoista luodaan linux:n shell skripti, joka on esitetty esimerkissä 5

```
# Luodaan tunnus, joka luodaan sivulta
# http://193.166.152.141/kirjautuminen.php
# Muuttujat otetaan kyseisen sivun kautta komentoriviargumenttina
# eli $1 on tunnuksen nimi, ja $2 on tunnuksen salasana.
adduser -g 501 -m -s /usr/lib/ssh/sftp-server $1

# tulostetaan kyseiset muuttujat luonti tiedostoon.
recho $1:$2 >> /bin/luonti

# Edellisessä vaiheessa luotu tiedosto tulostetaan niin,
# että kyseinen tuloste käytetään chpasswd sovelluksessa,
# joka siis määrittelee käyttäjälle salasanan.
cat /bin/luonti | chpasswd

# Luodaan käyttäjäkohtaiset hakemistot, julkinen www-hakemisto ja
# biisilistoja varten oma hakemisto "listat". Lisäksi
# kopioidaan käyttäjän kotisivu.

mkdir /home/$1/public_html
cp /var/www/tiedostot/index.html /home/$1/public_html/
mkdir /home/$1/listat

# Luodaan MP3 hakemisto, jonne käyttäjällä on oikeus uppia
# omat mp3 tiedostonsa. Lisäksi asetetaan käyttäjän kotihakemisto
# mp3 hakemistoon.

mkdir /home/$1/mp3
chown $1:muser /home/$1/mp3

# Lopuksi poistetaan luonti niminen tiedosto.
rm -f /bin/luonti
```

Esimerkki 5. Käyttäjänluonti-skripti

Skriptiä (esimerkki 5) käytetään PHP:n funktiolla *exec*. Tällä tavoin toimittuna tunnuksen luonti tapahtuu verkkopalvelusta, mutta saattaa aiheuttaa ongelmia, jos käyttäjän syöttämiä tietoja ei valvota. /6, 7/

5.2.2 Tunnuksen luonti PHP:lla

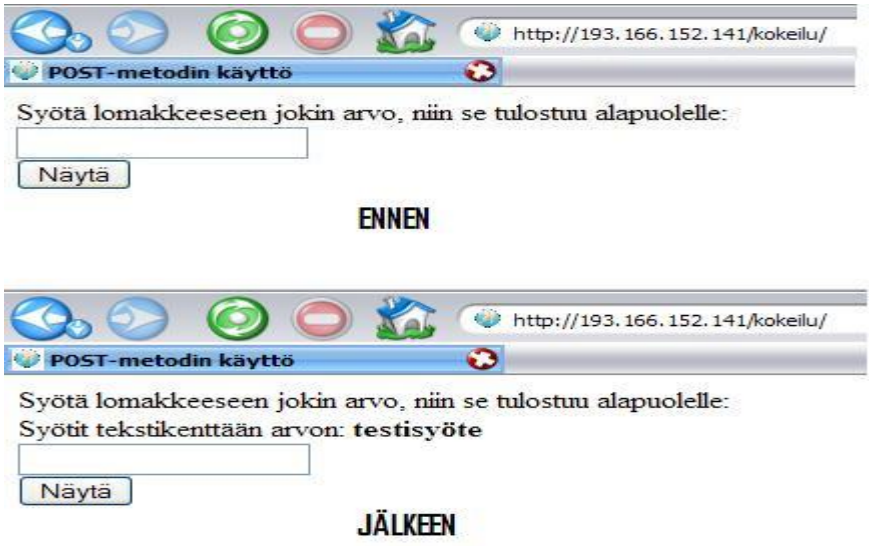
Tarkoituksena oli, että käyttäjä luo tunnuksensa verkkopalvelun kautta. Perusideana halusimme käyttää html:n lomakkeiden *POST*-metodia tiedon välittämiseen eteenpäin. PHP:n osuus oli tutkia tullutta syötettä ja tehdä päätöksiä ehtolauseiden avulla. Menetelmä oli siis teoriatasolla hyvin yksinkertainen, mutta toteu-

tuksessa tuli ottaa huomioon monia asioita. Esimerkissä 6 on luotu hyvin yksinkertainen sivu, jossa PHP tulostaa tekstikenttään syötetyn arvon.

```
<html>
<head>
<title>POST-metodin käyttö</title>
</head>
<body>
Syötä lomakkeeseen jokin arvo, niin se tulostuu alapuolelle:<br />
<?
//Jos on lähetetty lomakkeen kenttä arvo niin se kaiutetaan
if(isset($_POST['arvo'])) {
echo "Syötit tekstikenttään arvon:<b> " . $_POST['arvo'] . "</b>";
}
?>
<!--Luodaan lomake, jossa on tekstikenttä arvo ja lähetysnappula-->
<form method="POST">
<input type="text" name="arvo" /><br />
<input type="submit" value="Näytä" />
</form>
</body>
</html>
```

Esimerkki 6. POST-metodia hyödyntävä sivu

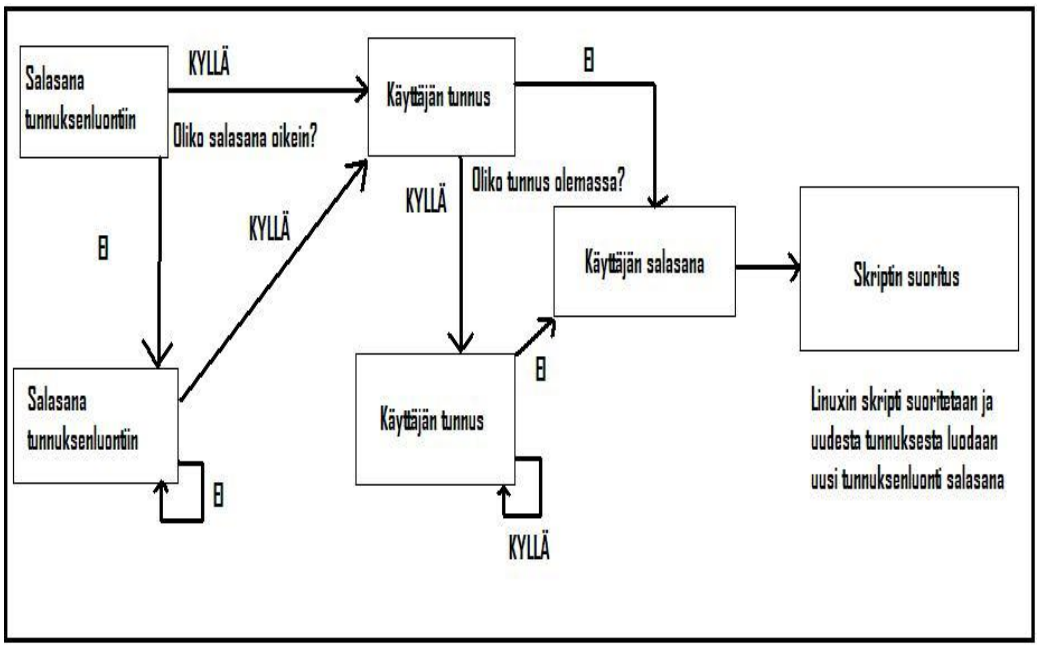
Esimerkin 6 toiminta on hyvin yksinkertainen, sillä siinä ei tarkkailla kuin yhtä arvoa. Varsinaisessa palvelussa kuljetetaan monia muuttujia samanaikaisesti eteenpäin. Esimerkin 6 toiminta selaimessa on esitetty kuvassa 13.



Kuva 13. Sivun kokeilu

Tunnuksien luonnissa ensimmäisenä huomioitavana asiana oli jo aiemmin mainitsemani oikeuksien rajaaminen, siitä kenellä on oikeus luoda tunnukset. Rajaus tapahtui siten, että käyttäjän oli tiedettävä salasana. Aluksi salasana oli vakiomerkkijono, mutta myöhemmin otimme käyttöön tietokannasta haetut salasanat. Ongelmana oli, että uusia salasanoja oli luotava samaan tahtiin kuin vanhoja käytettiin. Päätimme ratkaista ongelman luonnollisella tavalla luomalla uuden salasanan samalla, kun uusi käyttäjä luodaan. Salasanan muodostimme uuden käyttäjän tunnuksesta muuttamalla se kymmenen merkin pituiseksi sha1:ksi. Näin menettelemällä saimme sopivan vaikeita salasanoja aikaiseksi. /18/

Varsinainen tunnuksen luonti tehtiin aiemmin tehdyllä skriptillä. Tietoja kerättiin skriptiin yksittäisillä lomakkeilla, joiden tietoa välitettiin eteenpäin *POST*-metodilla. Käyttäjän tunnusta halusimme tarkkailla siten, että varmistimme tietokannasta tunnuksen olevan uusi. Käyttäjän salasanaa emme tarkkailleet, sillä oletimme käyttäjän osaavan luoda hyvän salasanan. Tunnuksen luonti tapahtui seuraavan kuvan 14 mukaisesti:



Kuva 14. Tunnuksen luonti

Tiedoston sisältö kokonaisuudessaan on Liitteessä 1. Kuvassa 14 muutamat kohdat, kuten salasana tunnuksenluontiin ja käyttäjän tunnus, ovat kahteen kertaan, koska ne ovat erillisissä ehtolauseissa. Niiden sisältö on lähes identtinen, mutta kuvassa 14 alempana olevissa lokeroissa tulee ilmoitus virheellisestä syötteestä. /18/

5.3 Käyttäjän omat sivut

Käyttäjien erottaminen toisistaan tapahtui omilla hakemistoillaan. Käyttäjän sivut sijoitettiin luonnollisesti *public_html*-hakemistoon, jonne pääsee myös selaimella. Sivuilla haluttiin alusta asti erottaa matkapuhelinkäyttäjä ja tietokoneen käyttäjä. Syynä haluun erottaa nämä kaksi ryhmää toisistaan oli se, että matkapuhelimen näyttö on hyvin paljon pienempi kuin tietokoneen. Erottelun perusteena päätimme käyttää selaimen nimeä. Tavallisesti tietokonekäyttäjällä selain on Firefox, IE tai Opera. Matkapuhelinpuolella selain kantaa puhelimen merkkiä mukanaan. Käytössämme oli Nokian puhelin. Tiedon käyttäjän järjestelmästä saa käyttämällä *$_SERVER['HTTP_USER_AGENT']*. Se palauttaa järjestelmän tiedot. Puhelimella selaimella sivuilla huomasimme ainakin Nokian puhelimen sisältävän merkkijonon SymbianOS. Tulosteena tuli myös paljon muuta tietoa, mutta halusimme vain kyseisen osan tunnistukseen. Se onnistui käyttämällä PHP:n funktiota *stristr*, joka aloittaa tiedon lukemisen kohdasta, jossa on haluttu merkkijono. Lopuksi loppuosa täytyi karsia pois, jottei esimerkiksi käyttöjärjestelmän versio tulisi mukaan. Se onnistui käyttämällä funktiota *substr* ja antamalla funktiolle sopiva pituusmäärite. /18/ Näin voidaan suhteellisen helposti erotella käyttäjän järjestelmän tyyppi.

Sivuille halusimme tilanteeseen sopivat toiminnot. Niitä olivat tässä tapauksessa soittolistojen luonti, niiden muokkaaminen ja poistaminen. Myöhemmin lisäsimme toiminnoiksi myös kappaleiden poiston ja niiden siirtämisen käyttäjän hakemistoista, jotka eivät sisältyneet http-palvelimen piiriin. Jälleen kerran toteutimme toiminnot osittain PHP:llä ja osittain Linuxin skripteillä.

Ennen kuin yhtään soittolistaa luotiin tarvitsi päättää niiden tallennusmuoto. Harkitsimme tietokantaan tallennettuja listoja ja puhtaasti tekstimuodossa olevat listat. Päädyimme käyttämään tekstimuotoisia listoja niiden siirrettävyyden ja helppouden vuoksi. Kaiken lisäksi emme halunneet tyytyä yhteen listatyyppiin, vaan loimme kolme erilaista listaa yhdestä. Erottelu oli tarpeen puhelimen ja tietokoneen listojen välillä, sillä tavalliset soittimet voivat toistaa suoraan pls-tyyppisiä listoja. Puhelimien listat laitoimme yksinkertaisempaan muotoon, jossa oli pelkästään kappaleen sijainti ja nimi muodossa *http://palvelin/~käyttäjä/kappale.mp3*. Kolmantena listatyyppinä olivat pelkät kappaleiden nimet sisältävä lista helpottamaan käsittelyä. Tieto olemassa olevista kappaleista ja listoista tallennettiin kukin omaan tiedostoonsa. Kaikkien kappaleiden nimet tallennettiin *biisit.txt* tiedostoon ja soittolistat *pls_lista.txt* tiedostoon. Aluksi nämä kaksi aiemmin mainitsemaani tiedostoa olivat *public_html*-hakemiston juuressa, mutta havaittuamme huomattavan tietoturvariskin siirsimme ne apachen ulottumattomiin *listat*-hakemistoon käyttäjän kotihakemiston taakse. Riski syntyi siitä, että jokaisella käyttäjällä olisi ollut suoraan selaimella luettavat tekstitiedostot, jotka olisivat sisältäneet kaikkien kappaleiden nimet. Kappaleiden nimillä ne olisivat olleet suoraan ladattavissa ja kuunneltavissa.

Huomasimme yhdessä vaiheessa, että käyttäjä ei voi itse suoraan siirtää kappaleita *public_html*-hakemistoon. Tämä havainto ei meitä haitannut, sillä se tarkoitti myös, että käyttäjä ei voinut muokata kyseisessä kansiossa olevia tiedostoja. Kappaleiden siirron omaan verkkohakemistoon toteutimme tästä johtuen Linuxin skriptillä ja loimme käyttäjälle kansion *mp3*, johon siirrettävät kappaleet tuli sijoittaa.

5.3.1 Käyttäjän sivujen sisältö

Suunnittelun jälkeen olimme valmiita siirtymään sisällön tuottamiseen. Halusimme jälleen käyttää aiemmin jo tunnuksenluonti-palvelussa käytettyä *POST*-

metodia. Tällä kertaa kyseistä metodia käytettiin myös sivuilla selailuun. Sivujen toiminta oli siis jälleen suhteellisen selkeä teoriatasolla. Rakenteen ainoa ongelma oli lähinnä pituus, sillä kyseisen tavan käyttö vaati monia ehtolauseita.

Ensimmäinen osa sivusta, jonka loimme, oli käyttäjän kappaleiden listaus. Kyseistä tehtävää varten olimme jo alustavasti suunnitelleet tietojen tallentamista *biisit.txt*-tiedostoon. Kyseinen tehtävä suoritettiin esimerkin 7 mukaisesti.

```
cd /home/$1/public_html/
find *.mp3 > /home/$1/listat/biisit.txt
```

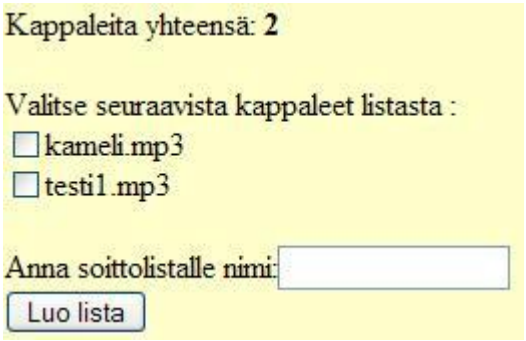
Esimerkki 7. Biisi.txt tiedoston luonti

Tietojen ollessa yhdessä tiedostossa oli helppo esittää ne sivuilla. Seuraava vaihe kaikkien kappaleiden esittämisessä oli käyttää PHP:tä tietojen lukemiseen ja esittämiseen. Tiedoston luku tapahtuu *file*-funktiolla. Se lukee tiedoston matriisiin(array), josta se on helppo esittää rivi kerrallaan. Sen käyttö on hyvin yksinkertaista: *file("/polku/tiedosto")*. Tiedostosta voidaan lukea sivulle tietoa rivi kerrallaan esimerkin 8 mukaisesti.

```
$bskripti="sudo /bin/biisit " . $usernimi;
exec("$bskripti");
$lines=file('/home/' . $usernimi . '/listat/biisit.txt');
echo "Kappaleita yhteensä:<b> ";
echo count($lines);
echo "</b><br />";
echo "<br />";
?>
<form action="" method="post">
Valitse seuraavista kappaleet listasta :<br />
<?
foreach ($lines as $line_num => $line) {

echo "<input type=checkbox name=biitit[] value=$line>$line</input><br />";
}
?>
<br />

Anna soittolistalle nimi:<input type="text" name="uuspls" size="15" />
<br />
<input type="submit" name="tokat" value="Luo lista" />

</form>
```

Esimerkki 8. Tiedoston sisällön listaus

Esimerkkiä 8 käytettiin myös soittolistan luonnin valmisteluun. Esimerkki 8 esittää kappaleiden lukumäärän, sekä jokaisen kappaleen valintalaatikkona (checkbox) ja tekstikentän, johon uuden listan nimi tulee.



Kuva 15. Soittolistan luonti

Lomakkeen (kuva 15) kautta saatu tieto välitetään Linuxin skriptille siten, että ensimmäinen komentoriviargumentti on käyttäjän tunnus, toinen on listan nimi ja kolmannesta eteenpäin kappaleiden nimiä. Näin ollen teimme skriptin (esimerkki 9) soittolistojen tekoa varten.

```
#!/bin/bash

# kappalenumero .pls tiedostossa
kappalenro=1

# alustetaan ensimmäiset argumentit niin, että se poimii
# käyttäjänimen ja tiedostonimen.

kayttaja=$1
tnimi=$2
maara=`expr $# - 2`
# Siirretään komentoriviargumentit tiedostoon siihen asti,
# että lopulta vastaan tulee tyhjä argumentti.
# Tässä skriptissä tehdään sekä osoitteellinen playlisti,
# että myös ilman osoitetta lukemisen helpottamiseksi.

# mobiilipäätteen playlist
recho http://193.166.152.141/~$kayttaja/$3 > /home/$kayttaja/public_html/$tnimi.mur

# pelkistetty kappalelista
recho $3 > /home/$kayttaja/public_html/$tnimi.plai

# tietokonesopivan playlist
recho [playlist] > /home/$kayttaja/public_html/$tnimi.pls
recho File$kappalenro = http://193.166.152.141/~$kayttaja/$3  >> /home/$kayttaja/public_html/$tnimi.pls
recho Title$kappalenro = $3 >> /home/$kayttaja/public_html/$tnimi.pls

while [ "$4" != "" ]; do

# lisätään yhdellä kappalenro:lla
   kappalenro=`expr $kappalenro + 1`

# matkapuhelin playlist teko
    echo http://193.166.152.141/~$kayttaja/"$4" >> /home/$kayttaja/public_html/$tnimi.mur
    echo "$4" >> /home/$kayttaja/public_html/$tnimi.plai

# tietokoneplaylistin teko
    echo File$kappalenro = http://193.166.152.141/~$kayttaja/$4  >> /home/$kayttaja/public_html/$tnimi.pls
    echo Title$kappalenro = $4 >> /home/$kayttaja/public_html/$tnimi.pls

# Siirtää yhdellä alaspäin
    shift

done
recho NumberOfEntries=$maara >> /home/$kayttaja/public_html/$tnimi.pls
recho Version=2  >> /home/$kayttaja/public_html/$tnimi.pls
```

Esimerkki 9. Soittolistan luonti skriptillä

Uutta esimerkissä 9 aiempiin verrattuna on *shift*. Se siirtää komentoriviargumentteja yhden alaspäin. Toimenpide on hyvin hyödyllinen, sillä sitä käytettäessä ei tarvitse tarkkailla kappaleiden kokonaismäärää. /17/ Otimme kyseisen skriptin luonnissa huomioon myös sen, että sitä pystyi käyttämään myös soittolistan muokkaamiseen.

Soittolistojen jälkikäsittely tapahtui hyvin samankaltaisesti kuin niiden luontikin. Erona uuden luonnilla ja vanhan muokkaamisella oli se, että olemassa olevien listojen nimiä ei enää muutettu ja niiden sisältö tuli myös esittää muokkaustilanteessa. Aiemmin luotujen listojen esittäminen tapahtui samalla tavalla kuin esimerkissä listaus. Ainoana erona oli käsiteltävä tiedosto, *pls_lista.txt*, johon oli listattu kaikki luodut soittolistat. Listojen käsittelyyn sopii kuitenkin paremmin yhden listan valinta kerrallaan. Niinpä käytimme lomakkeessa radionappuloita ja

yhden lähetyspainikkeen sijaan teimme niitä kolme kappaletta, joilla kullakin oli oma tehtävänsä. Tarvittavat tehtävät olivat listojen tarkastelu, muokkaus ja poisto. Tarkastelunapin toiminta on samanlainen kuin soittolistojen listaus tai kappaleiden listaus. Avattava tiedosto on tässä tapauksessa soittolistan plai-tiedostomuoto. Soittolistan poisto tapahtuu yksinkertaisen skriptin avulla, joka sisältää vain komennon *rm –f soittolista*. Soittolistoja on samalla nimellä kolme kappaletta: *plai-*, *pls-* ja *mur*-päätteiset tiedostot. Rivejä skriptissä on siis myös kolme kappaletta. Soittolistan muokkauksessa tilanne oli hyvin samankaltainen kuin listaa luotaessa siitä johtuen kaikki kappaleet listataan. Erona muokkauksessa on, että kappaleet jotka jo ovat listalla näkyvät tummennettuna ja valittuna. Tämä toteutettiin PHP:n funktiolla *in_array* (kts. esimerkki 10), jota käytettiin if-lauseen valintaperusteena.

```
if(in_array($kpl, $listasis)){
echo "<input type=checkbox name=biitit[] value=$kpl checked><b>$kpl</b></input><br />";

}
```

Esimerkki 10. Ennestään listalla olevien kappaleiden tunnistaminen

Esimerkissä 10 muuttuja *$kpl* esittää kyseisellä linjalla olevaa kappaletta kaikkien kappaleiden joukosta. Muuttuja $listasis edustaa valitun soittolistan matriisia. Kappaleiden valinnan jälkeen listan luonnissa käytetään samaa skriptiä kuin esimerkissä 9.

Kuva 17. Soittolistojen hallinta

Kuvassa 17 esitetyt toiminnot takaavat listojen sujuvan käsittelyn. Niiden lisäksi tarvittiin vielä toiminnot kappaleiden siirto *public_html*-hakemistoon sekä niiden poistamiseen sieltä.

Kappaleiden siirto haluttiin toteuttaa mahdollisimman yksinkertaisesti. Yksinkertaisin tapa oli kopioida kaikki kappaleet *mp3*-hakemistosta. Tarvittiin siis vain yksinkertainen skripti, joka ensin kopioi kappaleet ja sen jälkeen poisti ne. PHP:n osuudeksi jäi vain suorittaa skripti (esimerkki 11).

```
#!/bin/sh
# siirretään käyttäjän X mp3:t public_html:ään
mv -f /home/$1/mp3/*.mp3 /home/$1/public_html
#poistetaan tiedostot kansiosta
rm -f /home/$1/mp3/*.mp3
```

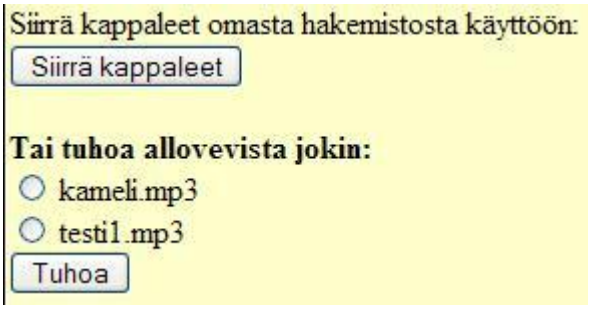
Esimerkki 11. Kappaleiden siirto

Kappaleiden poistoon *public_html*-kansiosta käytettiin yksinkertaista skriptiä (esimerkki 12), johon komentoriviargumenteiksi tuli vain käyttäjätunnus ja poistettava kappale.

```
#!/bin/sh
# Tuhoaa valitut mp3:set sivustolla
rm -f /home/$1/public_html/$2
```

Esimerkki 12. Kappaleiden poisto

PHP:ssä skriptiin valittiin kaikkien kappaleiden joukosta yksi poistettava kappale radionappulan avulla. Tilanne on esitetty kuvassa 18.



Kuva 18. Kappaleiden siirto ja poisto

5.3.2 Kirjautuminen omille sivuille

Viimeisenä vaiheena oli suunnitella tapaa, jolla käyttäjä kirjautuisi sivuillensa. Harkitsimme aluksi, käyttäisimmekö siihen sessioita. Sessioiden etuina oli käyttäjän helppo autentikointi ja auktorisointi. Halusimme kuitenkin toteuttaa käyttäjän todentamisen hieman erilaisella tavalla, käyttämällä käyttäjän IP-osoitetta. Syy kyseiseen valintaan johtui mahdollisista jatkokäyttötarkoituksista ja seurannan mahdollisuudesta.

Ideana oli ensin tallentaa käyttäjän IP-osoite tietokantaan. Omille sivuille siirryttäessä osoitetta verrattaisiin samalla rivillä olevaan käyttäjätunnukseen. Tarkoitusta varten sopiva taulu (kuva 19) oli jo olemassa, *user*.

kayttaja	salasana	luotu	osoite
reiska	reiska	2007-05-10 22:39:16	80.223.79.15

Kuva 19. Käyttäjä kirjautuneena

Taulussa (kuva 19) on myös luotu-sarake, jonka mahdollinen käyttötarkoitus oli käyttäjän kirjaaminen ulos tietyn ajan jälkeen. Sarake oli *now*-tyyppinen, mikä tarkoittaa sen päivittyvän aina, kun rivillä tapahtuu muutos.

Kirjautuminen palveluun tapahtui varsin perinteisesti, eli lomakkeella, jossa oli tunnus- ja salasanakentät sekä lähetysnappula kuvan 20 mukaisesti.

Kuva 20. Kirjautuminen

PHP:hen tarvitsi määritellä käyttäjätunnuksen ja salasanan tarkastus, joka suoritettiin tietokannasta. Yleisesti ottaen ei ole tarpeen kertoa, jos käyttäjätunnus on oikein, mutta salasana ei. Todellisten käyttäjien tietäminen voi olla jollekin hakkerille houkutin kokeilla murtautua sisään. Me kuitenkin halusimme selvästi erottaa, oliko tunnus ja salasana oikein. Käyttäjätunnuksen ja salasanan tarkastuksessa on looginen rakenne. Aluksi otetaan lomakkeesta saadut tiedot muuttujiin, haetaan tietokannasta varmasti oikeat arvot toisiin muuttujiin ja verrataan niitä keskenään. Lopuksi tarkastellaan ehtolauseiden avulla, minne haaraan kuuluu edetä. Ehtolauseiden rakenteen voi ilmaista lyhyesti esimerkin 13 muodossa.

```
if(($otunnus == $stunnus) && ($osalasana == $ssalasana)){
Päivitä tietokantaan käyttäjän osoite
siirrä käyttäjä omille sivuillensa
}
elseif {
if($otunnus == $stunnus) {
Tallenna oikea tunnus käyttäjänimen arvoksi
Ilmoita väärästä salasanasta
}
elseif($otunnus != $stunnus) {
Ilmoita, ettei tunnusta ole olemassa
}
}
```

Esimerkki 13. Kirjautumisen ehtolauseet

Esimerkissä 13 *$otunnus* ja *$osalasana* tarkoittavat tietokannasta haettuja arvoja ja *$stunnus* ja *$ssalasana* taasen lomakkeeseen syötettyjä arvoja.

Viimeiseksi tuli vielä käyttäjän omille sivuille määritellä tarkastus tietokannan taulussa olevan IP-osoitteen ja käyttäjän nykyisen IP-osoitteen välille, jotta sivuille ei pääsisi suoraan. Vertailua varten tietokannasta haettava osoite saadaan ottamalla sivun osoitteesta se osa, jossa on käyttäjän nimi. Kyseinen toimenpide onnistuu käyttämällä PHP:n funktiota *substr*. Se ottaa merkkijonosta halutun osan. Sivuille tulijan osoite saadaan PHP:n ennalta määritellyllä muuttujalla *$_SERVER['REMOTE_ADDR']*. Näitä kahta aiemmin mainittua osoitetta vertailemalla tiedetään onko käyttäjä kirjautunut sisään.

Uloskirjautumisen periaate on melko sama kuin sisäänkirjautumisen yhteydessä, sillä silloin tietokannasta poistetaan käyttäjän osoitetieto. Tauluun sijoitettiin selvennyksen vuoksi arvo *NOT LOGGED IN*. Näin ollen olisi helppo seurata, ketkä ovat kirjautuneena sisään.

5.3.3 Käyttäjän sivujen lopullinen muoto

Kaikki aiemmin mainitut osat yhdistettiin lopulta sivuille (kts Liite 2). Viimeisenä tehtävänä oli vain lajitella tieto ymmärrettävään muotoon sivulle. Selkeys saatiin aikaiseksi lajittelemalla eri toiminnot eri painikkeiden alle. Navigointia varten oli luotu jo aiemmin selvä jaottelu sivun rakenteeseen. Lisäyksenä aiempiin toimintoina laitoimme sivuille myös mahdollisuuden laittaa ohjeistuksia käyttöä varten. Pienenä hienosäätönä asetimme myös logon sivuille ja painikkeita muutettiin luomalla tyylisivulla niiden ulkoasua erilaiseksi.

Kuva 21. Navigointi käyttäjän sivuilla

Lopputulos (kuva 21) oli suhteellisen suoraviivainen ja selkeästi etenevä kokonaisuus. Puutteita löytyi joistakin kohdista, kuten tilanteessa jolloin käyttäjä poistaa kappaleen, joka on jollain soittolistalla. Tällöin soittolistaan jää viittaus kappaleeseen, jota ei ole olemassa. Emme kuitenkaan pyrkineet täydellisyyteen, vaan todellisuutta jäljittelevään tilanteeseen.

6 RTSP-PALVELINOHJELMISTOT

RTSP ja RTMP on tarkemmin kuvattu teoriataustaa-osiossa. Kohdattuamme matkapuhelimen kanssa ongelmia streamauksessa http:n päällä, päätimme kokeilla

palvelinohjelmistoja, jotka pystyivät lähettämään tietoa RTSP-protokollan päällä. RTSP eroaa http:stä siinä, että sillä ohjataan pakettien numerointia ja järjestystä. Etuna pakettien järjestämisessä on siirron laadun takaaminen. Eri ohjelmistojen välillä on eroja tuetuissa tiedostomuodoissa ja lähetystyylissä. Lähetystyylejä on kaksi. Ne ovat *on-demand* ja *broadcast*. Erona tyyleillä on se, että käyttäjä ei pysty vaikuttamaan *broadcast*-tyyppiseen lähetykseen, vaan se toimii kuten radio. Toinen tyyli *on-demand* on täysin käyttäjän hallittavissa. /8, 9, 12/

Vastaamme tuli muutama varteenotettava vaihtoehto, joita päätimme kokeilla. Vaihtoehtoina olivat *Live555*, *DarwinStreamingServer* ja *Flash Media Server2*. Kokeilimme jokaista vaihtoehtoa hieman ja tutkimme niiden heikkouksia ja vahvuuksia. Tutkimustemme perusteella halusimme tehdä tilanteeseen parhaiten sopivan ratkaisun. Tilanteessamme paras ratkaisu oli ohjelmisto, joka toimisi pyynnöstä (*on-demand*) ja tiedostomuotona mielellään mp3.

6.1 Flash Media Server2

Palvelinohjelmiston pääasiallisena tarkoituksena on tarjota reaaliaikaisia streameja pääosin flv-tiedostoille. Se pystyy kuitenkin välittämään myös ääntä ja videota streamina. Etenkin tuki *on-demand* mp3:ille oli suuri etu ajatellen projektiamme. FMS2 (Flash Media Server2) käyttää kahdesta muusta palvelinohjelmistosta poiketen RTMP (Real Time Messaging Protocol):tä streamin laadun takaamiseen. RTMP ei sisällä hyötydataa, vaan on pysyvä yhteys asiakkaan ja palvelimen välillä. /13, 19/

Hyvistä ominaisuuksistaan huolimatta FMS2 vaati ympärilleen monia sellaisia tekniikoita, joita nykyiset matkapuhelimet eivät tue. FMS2 vaatii asiakasohjelmalta toimiakseen Flashplayer:n version 8 tai pikemminkin sen sisältämän actionscriptin (2.0). Matkapuhelimien sisältämä *flash lite 2.1* sisältää flashplayer:n version 7 ja ominaisuuksien mukaan myös actionscript 2.0:n. Tieto on kuitenkin harhaanjohtava, sillä flash lite:n actionscript ei sisältänyt FMS2:n käyttämiseen tarvittavia etäkäyttökomponentteja (remote components). Toteutusta oli siis mahdotonta edes

harkita tehtäväksi käyttäen FMS2:sta. Toinen paha kompastuskivi FMS2:n käytössä olisi ollut hinta, joka oli 4500$. /13/

6.2 Darwin Streaming Server (DSS)

Darwin on Applen avoimen lähdekoodin versio Quick Time Streaming Serveristä (qtss). DSS on perinteinen streaming-palvelin siinä mielessä, että se luottaa RTSP:hen. Asentaessamme ohjelmistoa huomasimme, että asentaminen urpm:n avulla ei toiminut. Ohjelmisto kyllä asentui, mutta sitä ei voinut käyttää. RPM paketin asentaminen onnistui mutkattomasti. DSS:n hyvänä puolena oli ehdottomasti selaimen kautta tapahtuva asetusten muuttaminen ja toistettavan sisällön hallinta. Pienen testailun jälkeen kuitenkin huomasimme, että ääntä pystyi lähettämään vain *broadcast*:na. Tämä oli paha takaisku, sillä DSS oli muuten näyttänyt lupaavalta vaihtoehdolta. Lisätietoja DSS:stä ja qss:stä löytyy osoitteesta http://developer.apple.com/opensource/server/streaming/qtss_admin_guide.pdf(15.5.2007). /8/

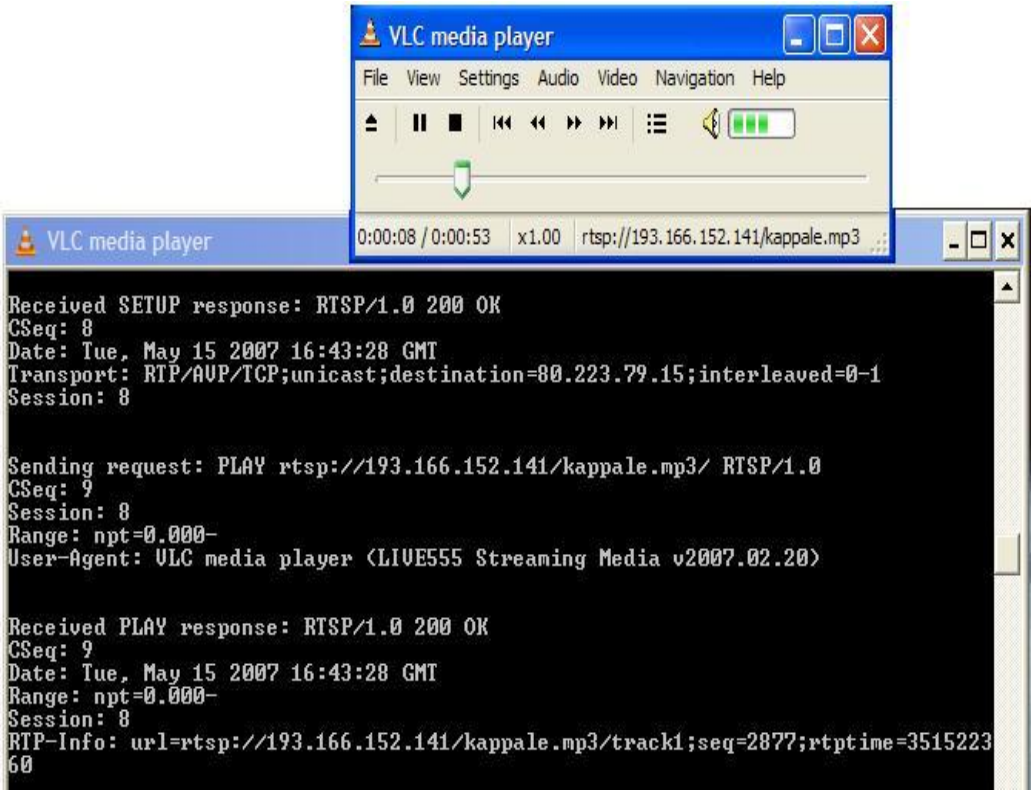
6.3 Live555

Live555 on kahteen aiempaan verrattuna hyvin pienikäyttöinen ja kompakti RTSP-palvelinohjelmisto. Sen käyttö on vielä yksinkertaisempaa kuin monien muiden vastaavien ohjelmistojen. Se yksinkertaisesti sijoitettiin esimerkiksi käyttäjien juurihakemistoon ja käynnistettiin (kuva 22) siellä. Tämän jälkeen Live555 ymmärsi RTSP-muotoiset pyynnöt ja toimi tiedon välittäjänä asiakkaille.

```
[root@mstream home]# ./live555MediaServer
LIVE555 Media Server
        version 0.15 (LIVE555 Streaming Media library version 2007.01.11).
Play streams from this server using the URL
        rtsp://193.166.152.141/<filename>
where <filename> is a file present in the current directory.
Each file's type is inferred from its name suffix:
        ".aac" => an AAC Audio (ADTS format) file
        ".amr" => an AMR Audio file
        ".m4e" => a MPEG-4 Video Elementary Stream file
        ".mp3" => a MPEG-1 or 2 Audio file
        ".mpg" => a MPEG-1 or 2 Program Stream (audio+video) file
        ".ts" => a MPEG Transport Stream file
                (a ".tsx" index file - if present - provides server 'trick play' support)
        ".wav" => a WAV Audio file
See http://www.live555.com/mediaServer/ for additional documentation.
```

Kuva 22. Live555 käynnissä

Ohjelmiston käyttö oli siis hyvin yksinkertaista, helppoa ja ennen kaikkea vähän resursseja varaavaa. Testasimme aluksi ohjelmiston toimintaa tietokoneella *VLC-playerin* avulla. Kyseisessä soittimessa on mukana myös kätevä debug-toiminto (kuva 23), jolla näkee palvelimen ja tietokoneen väliset sanomat.



Kuva 23. VLC ja Debug

Pelkän äänen kuulemisen lisäksi oli hyvä havaita, että myös RTSP oli täysin toiminnallinen. Tietokoneella testaamisen jälkeen oli aika siirtyä testailuun puhelimella. Ongelmat alkoivat heti puhelimen kanssa, sillä emme huomanneet S60 3rd-alustan tukevan eri tiedostomuotoja rtsp:llä. Onneksemme Live555 tuki myös amr-tiedostoja, joita myös puhelimme (N80) tukee RTSP:n päällä. Tiedostomuodon muutoksen jälkeen soitto tuli joskus streamina ja joskus taas se ei lähtenyt ollenkaan soimaan. Päätimmekin siis siirtyä takaisin puhtaasti http:n päälle.

7 LISÄARVOA TUOVAT UUDET OMINAISUUDET

Sivusto oli lopulta saatu haluttuun muotoon ja sen toimintaa oli testattu tavallisella tietokoneella ja kahdella yleisemmällä selaimella, Firefox:lla ja IE:llä. Vasta jälkeenpäin tutkiessa havaitsimme asioita, joita olisi voinut tehdä eri tavalla tai toteuttaa toisin. Käytännön kokemus on tärkeää uuden palvelun luonnissa. Meille tämä oli ensimmäinen luomamme palvelu ja siksi emme huomanneet katsoa asioita jokaiselta kannalta.

Eräs ehkä huomattavin parannusmahdollisuus olisi ollut tietoturvassa. Käyttäjän kappaleet ja soittolistat olivat suoraan saatavilla kaikille. Tiedostojen piilottamiseen oli monta mahdollista keinoa, mutta tehokkain niistä olisi ollut sisällyttää ne tietokantaan *BLOB*:na (Binary Large Object) /11/. Kyseistä menettelytapaa käyttäen tiedostoihin pääsyyn olisi vaadittu erillinen rajapinta tietokantaan. Tietokoneissa ongelmaa ei esiinny, mutta matkapuhelimien rajattu suorituskyky vaatii sen. Rajapinta suorittaisi kyselyt tietokannasta ja palauttaisi tiedon puhelimelle selvästi ymmärrettävässä muodossa. /11/

Toinen hyvin merkittävä parannus olisi ollut pääkäyttäjän omat sivut. Niiden tarkoituksena olisi ollut hallita ja valvoa niin palvelimen kuin käyttäjienkin toimia selaimen kautta. Sivuilla olisi pystynyt poistamaan käyttäjiä, seuraamaan ketkä

ovat kirjautuneena sisälle ja tarkkailla logi-tiedostoja. Lisäksi mahdollisuus päivittää kaikkia sivuja, kuten lisätä tiedotteita, olisi luonut helppouden tunteen.

Lopullisessa tuotoksessa käyttäjillä oli loputon määrä levytilaa käytössä, mikä olisi oikeassa tilanteessa erittäin huono asetelma väärinkäytösten kannalta. Ongelman saisi korjattua käyttämällä *quota*:a. Sillä voidaan määritellä käyttäjäkohtainen levyn käyttö. Lisähyödyn rajauksesta saisi määrittelemällä eritasoisille käyttäjille erisuuruisia levynkäyttöoikeuksia.

Palvelinta olisi voinut automatisoida luomalla sinne myös sähköpostipalvelimen. Sähköpostipalvelin olisi kyennyt lähettämään käyttäjälle tiedon salasanasta tai kenties viikoittaisen uutiskirjeen. Esimerkiksi uuden käyttäjän luonnin yhteydessä tiedot tilistä menisivät automaattisesti käyttäjälle itselleen tai salasanan unohtuessa uusi toimitettaisiin määriteltyyn osoitteeseen. Helppo ohjelma sähköpostin lähetykseen on esimerkiksi *SendMail*, joka toimii komentoriviltä. Lisäksi sitä voisi käyttää käyttäjien sisäiseen viestintään.

8 PALVELUN TULEVAISUUDEN NÄKYMÄT

Uusia palveluita kehitettäessä on myös hyvä miettiä, mikä on sen suhde käytännön tarpeisiin. Hyvä idea yksinään on harvoin kauaskantava. Uuden yhdistäminen aiemmin hyväksi todettuihin asioihin tuottaa helpommin ja pienemmällä vaivalla hyvän tuloksen. Sama tilanne koski myös luomaamme palvelua. Pelkkä toiminnallisesti yksinkertaisesti palvelu tuskin houkuttelisi käyttäjiä, vaikka monia vastaavankaltaisia palveluja ei olekaan olemassa.

Eräs hyvin tärkeä asia loisti poissaolollaan. Se oli monipuolisuus. Käyttäjän oma musiikkivarasto saattaa olla hyvinkin pieni ja siksi uusien kappaleiden hankinnan voisi yhdistää palveluun. Kappaleiden ollessa hyvässä tallessa palvelimella myös

tekijänoikeudet olisivat paremmin turvattuna. Palvelun tarjoaja olisi vastuussa tiedon säilymisestä, eikä varmuuskopioita tarvitsisi ottaa. Käyttäjä ei varsinaisesti omistaisi kappaletta, vaan se olisi vuokralla.

MP3:ten streamaus vaatii toimiakseen tavallista gprs-kaistaa suuremman nopeuden (n. 128kb/s). Nopeampien yhteyksien ongelmana on niiden hinta. Hinnat ovat kuitenkin olleet laskussa ja tällä hetkellä esimerkiksi Saunalahden mobiililaajakaistojen hinnat ovat halvempia kuin vastaavilla nopeuksilla toimivat tavalliset laajakaistat (http://saunalahti.fi/tiedote/tiedote.php?index=2673 17.5.2007).

9 PROJEKTIN TULOKSET JA POHDINTA TYÖSTÄ

Alkuperäisenä tavoitteena oli saada aikaiseksi palvelinympäristö, jonka avulla oli mahdollista tarjota matkapuhelinsovellukselle musiikkia streamina. Musiikin muotona tuli olla mp3 ja siirtoprotokollana http.

Välillä kokeilimme eri tiedostomuotoja ja siirtoprotokollia. Lopulta kuitenkin palasimme jälleen alkuperäiseen suunnitelmaan sillä erotuksella, että musiikin tiedostomuoto oli amr. Kyse ei ollut siitä, ettei olisi ollut mahdollista soittaa mp3-muotoista musiikkia streamina http:n kautta, vaan siitä että olimme valinneet matkapuhelinten alustaksi Symbianin. Jos valintamme olisivat olleet Sony-Ericssonin uudet mallit, niin myös progressiivinen streamaus mp3:lla olisi onnistunut. Kyseinen asia on myös syy siihen, että emme muuttaneet palvelimella käsittelyä mp3:sta amr:än. Muutos ei olisi ollut suuri, mutta merkityksetön. Tavoite, jota emme pystyneet täyttämään, oli Java-sovelluksen laitteistoriippumattomuus. Siihen vaikutti jo aiemmin mainitsemani puhelimen tarjoamat järjestelmän palvelut Java-sovellukselle.

Palvelimen asennus ja sen käyttökuntoon saattaminen oli melko yksinkertaista ja suoraviivaista. Vaikein ja aikaa vievin osuus oli sivustojen luonti. Niiden toteutus oli melko suoraviivaista, kun vain ensiksi oivallettiin, mitä tehdä. Pohdintaan ja

kokeiluihin menikin lukemattomia tunteja. Aina uutta kohtaa tehtäessä oli mietittävä, miten se vaikuttaa muihin osiin ja miten paljon sitä pystyi jälkeenpäin muuttamaan.

Omalta kannaltani katsottuna tärkeintä ei ollut lopputulos, vaan se, mitä työnaikana tuli opittua. Monet ennen työn aloittamista vaikealta tuntuneet asiat muuttuivat loogisiksi käytännön kautta.

10 LÄHTEET

1. Apache Software Foundation 2006. Apachen moduulit [Verkkodokumentti] [Viitattu 20.5.2007] Saatavissa: http://httpd.apache.org/docs/2.0/mod

2. Apache Software Foundation 2006. Apachen ohjaimet [Verkkodokumentti] [Viitattu:20.5.2007]Saatavissa:
http://httpd.apache.org/docs/2.0/quickreference.html

3. Apache Software Foundation 2006. Apachen autentikointi [Verkkodokumentti] [Viitattu 20.5.2007] Saatavissa: http://httpd.apache.org/docs/2.0/howto/auth.html

4. Apache Software Foundation 2006. Moniprosessointimoduulit [Verkkodokumentti].[Viitattu:20.5.2007].Saatavissa:
http://httpd.apache.org/docs/2.0/mpm.html

5. Apache Software Foundation 2006. prefork mpm apachessa [Verkkodokumentti].[Viitattu:20.5.2007]Saatavissa:
http://httpd.apache.org/docs/2.0/mod/prefork.html

6. Koski R. 2000. Linux – käyttäjän käsikirja. Jyväskylä. Oy Edita Ab

7. Koski R. & Tervo E. 1999. Tehokäyttäjän opas- Linux. Jyväskylä. Suomen Atk- kustannus Oy

8. Tuomola K. & Lähdesmäki M. 2004. Streaming Media. Lapin yliopisto. Taiteiden tiedekunta. [Verkkodokumentti] [Viitattu 20.5.2007] Saatavissa: http://www.digmo.fi/digmore/Streaming.pdf

9. Nokia Corporation. 2007. Symbian 60 Nokian matkapuhelimissa. [Verkkodokumentti] [Viitattu 20.5.2007] Saatavissa: http://sw.nokia.com/id/2019b327-7e01-49c1-a60a-4dc48da8a105/S60_Platform_FAQ_v1_9_en.pdf

10. Nokia Corporation. 2006. Symbian 60 alustan perusteet. [Verkkodokumentti] [Viitattu: 20.5.2007] Saatavissa: http://sw.nokia.com/id/bceaffad-1807-43b6-9cd9-8519b4794b5c/S60_Platform_Basics_v1_0_en.pdf

11. MySQL AB. 2007. MySQL:n viiteohjekirja. [Verkkodokumentti] [Viitattu 20.5.2007] Saatavissa: http://downloads.mysql.com/docs/refman-4.1-en.a4.pdf

12. Vuorimaa, P. 2006. Multimediatekniikka. [Verkkodokumentti] [Viitattu: 20.5.2007].Saatavissa:http://www.tml.tkk.fi/Opinnot/T-111.2350/2006/Mmtekn_0_1.pdf

13. Flash Media Server2.0:n verkkosivut. [Verkkodokumentti] [Viitattu: 20.5.2007] Saatavissa: http://www.adobe.com/products/flashmediaserver/

14. Vivek, G. Gite. 2002. Linux Shell Scripting Tutorial v1.05r3 A beginners handbook. – Linuxin perusteet [Verkkodokumentti] [Viitattu: 20.5.2007] Saatavissa: http://www.freeos.com/guides/lsst/misc.htm#unix

15. Vivek, G. Gite. 2002. Linux Shell Scripting Tutorial v1.05r3 A beginners handbook. – Shell skriptit [Verkkodokumentti] [Viitattu: 20.5.2007] Saatavissa: http://www.freeos.com/guides/lsst/ch01sec09.html

16. Vivek, G. Gite. 2002. Linux Shell Scripting Tutorial v1.05r3 A beginners handbook. – Shell aritmetiikka [Verkkodokumentti] [Viitattu: 20.5.2007] Saatavissa: http://www.freeos.com/guides/lsst/ch02sec07.html

17. Vivek, G. Gite. 2002. Linux Shell Scripting Tutorial v1.05r3 A beginners handbook. – While silmukat [Verkkodokumentti] [Viitattu: 20.5.2007] Saatavissa: http://www.freeos.com/guides/lsst/ch03sec07.html

18. Achour, M. & Betz, F & Dovgal, A. & Lopes, N. & Olson, P. & Richter, G. & Seguy, D. & Vrana, J. 2007. PHP Manual – funktiot kategorioittain. [Verkkodokumentti] [Viitattu: 20.5.2007]. Saatavissa: http://www.php.net/manual/en/extensions.php

19. Wikipedia. 2007. RTMP. [Verkkodokumentti]. [Viitattu 20.5.2007]. Saatavissa: http://en.wikipedia.org/wiki/Real_Time_Messaging_Protocol

20. Laaksonen, A. 2003. Käytännön PHP-opas. [Verkkodokumentti] [Viitattu: 21.5.2007] Saatavissa: http://www.ohjelmointiputka.net/oppaat/phpj.zip

21. Net Site Story. 2007. Mikä on Flash?. [Verkkodokumentti] [Viitattu: 21.5.2007] Saatavissa: http://www.netsitestory.com/Flash/index.html

22. 2K mediat.com. 2007. JSP: Java Server Pages. [Verkkoartikkeli] [Viitattu: 21.5.2007] Saatavissa: http://www.2kmediat.com/internetohjelmointi/jsp.asp

LIITE 1

LIITTEET

LIITE 1 Käyttäjän kirjautumissivusto

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Rekister&ouml;ityminen palveluun</title>
</head>

<body>
<table width="90%" height="90%" border="1" cellpadding="2" cellspacing="5">
  <tr>
    <th width="20%" height="251" bgcolor="#33CCFF" scope="row"><p>LOGO</p>
      <p> </p>


  <p>navigointialue</p>

     <form id="kirjautuminen" name="kirjautuminen" method="POST" action="sivu2.php">
       <label>kayttajatunnus
       <input name="tunnus" type="text" id="tunnus" />
       </label>
          <p>
            <label>salasana <br />
            <input name="salasana" type="password" id="salasana" />
            </label>
       </p>
          <p>
            <label>
            <input type="submit" name="Submit" value="Kirjaudu" />
            </label>
          </p>
     </form>
     <p> </p>
     <p>tunnuksien luonti PHP?!?</p>
    <p>uloskirjaus </p>


    </th>
```

LIITE 1

```
   <td width="70%" bgcolor="#FFFFCC"><p> </p>
                <?php
                $conn = mysql_connect('193.166.152.141', 'antti', 'tekpo03') or die
('Virhe yritättäessä yhdistää');
mysql_select_db('muser');



if ($_SERVER['REQUEST_METHOD']=='GET') {
?>
   <p>T&auml;m&auml; on Villen ja Antin opinn&auml;ytety&ouml;</p>
   <p>tarvitset kirjautuakseen erilliset tunnukset</p>
   <p>Sy&ouml;t&auml; tunnus:</p>
   <form id="form1" name="form1" method="post" action="kirjautuminen.php">
     <label>tunnus
     <input type="password" name="syotto" />
     </label>
     <br />

     <label>rekisteröitymiseen
       <input type="submit" name="Submit" value="Jatka" />
       </label>
   </form>
<?


}

elseif(isset($_POST['syotto'])) {
$kirjtun = $_POST['syotto'];
$query = "SELECT koodi from tunnarit where koodi='$kirjtun'";
// myöhemmin voi lisätä kyselyn, joka voisi olla muotoa:
//"SELECT koodi from tunnarit where koodi = '$_POST['yritetty_tunnus']'";
//tulee lisätä tapahtumaan vasta post toiminnon jälkeen
$haettu = mysql_query($query) or die('Ei onnaa');
$tulos = mysql_fetch_row($haettu);
if($kirjtun == $tulos[0]) {
?>
<form action="kirjautuminen.php" method="post">
Anna käyttäjätunnus: <input name="tunnus" type="text" />
<input name="syotto2" type="hidden" value="<? echo $kirjtun ?>" /><br />
<input name="" type="submit" value="Jatka" />
</form>
<?
}
elseif($kirjtun != $tulos[0]) {
?>
Tunnus ei täsmännyt!<br />
Yritä uudelleen<br />
<form action="kirjautuminen.php" method="post">
```

LIITE 1

```
syötä tunnus: <input name="syotto" type="text" /><br />
<input name="" type="submit" value="Jatka" />
</form>
<?
}
}
if(isset($_POST['tunnus'])) {
$query2 = "SELECT kayttaja FROM `user` where kayttaja = '$_POST[tunnus]'";
$tarkistetaan = mysql_query($query2) or die('Haussa häikkää');
$tulos2 = mysql_fetch_row($tarkistetaan);
if($tulos2[0] != $_POST['tunnus']) {
?>
Tunnuksesi: <? $_POST['tunnus'] ?> <br />
Anna tunnuksellesi salasana: <form action="kirjautuminen.php" method="post">
<input name="salasana" type="password" />
<input name="syotto2" type="hidden" value="<? echo $_POST['syotto2'] ?>" />
<input name="tunnus2" type="hidden" value="<? echo $_POST['tunnus'] ?>" />
<input name="" type="submit" value="Jatka" />
</form>
<? }
elseif($tulos2[0] == $_POST['tunnus']) {
?>
Tunnus <? echo $_POST['tunnus'] ?> on jo olemassa!<br />
Koita uudestaan käyttäjätunnusta:<form action="kirjautuminen.php" method="post">
<input name="tunnus" type="text" />
<input name="syotto2" type="hidden" value="<? echo $_POST['syotto2'] ?>" />
<input name="" type="submit" value="Jatka" />
</form>
<? }
}

elseif(isset($_POST['salasana'])) {

$tunnus1 = $_POST['tunnus2'];
echo "<br>";
$salasana1= $_POST['salasana'];
// $teksti="$tunnus1:$salasana1 \n";
// $tiedosto="luonti";
// $fp = fopen($tiedosto,"w");
// fwrite ($fp,$teksti);
// fclose ($fp);
// chmod($tiedosto,777);

// $teksti="$tunnus1";
// $tiedosto1="tunnusluonti";
// $fp = fopen($tiedosto1,"w");
// fwrite ($fp,$teksti);
// fclose ($fp);
$muuttuja3="sudo /bin/scripti1 " . $tunnus1 . " " .  $salasana1;
```

```
exec("$muuttuja3");
// chmod($tiedosto1,777);

$query3 = "INSERT INTO user (kayttaja, salasana) VALUES ('$tunnus1',
'$salasana1')";
mysql_query($query3) or die('Ei onnaa');
$koodattu = substr(sha1($tunnus1),0,10);
//uusi kirjautumistunnus tunnarit tauluun
$uuskirjtun = "INSERT INTO tunnarit (koodi) VALUES ('$koodattu')";
mysql_query($uuskirjtun);
//Lopuksi voisi vielä suorittaa aiemman tunnuksienluonti tunnuksen poiston:
//"delete from tunnarit where koodi = 'aiempi_tunnus'"
$entinenkoodi = $_POST['syotto2'];
$vanha = "DELETE from tunnarit WHERE koodi = '$entinenkoodi'";
mysql_query($vanha);
echo "Tunnus $tunnus1 on luotu.";
echo "<a href=http://193.166.152.141/sivu2.php>Takaisin pääsivulle</a><br />";
}


mysql_close($conn);
 ?>  <p>    </p></td>
 </tr>
</table>
</body>
</html>
```

LIITE 2 Käyttäjän omat sivut

```
<?
$conn = include '/var/www/html/tun.php';

            mysql_select_db('muser');
$tulo = $_SERVER['REQUEST_URI'];
$usernimi = substr($tulo, 2, -1);

$oso = $_SERVER['REMOTE_ADDR'];
$query = "SELECT osoite FROM user where kayttaja = '$usernimi'";
$haku = mysql_query($query) or die('Haussa häikkää');
$tulos2 = mysql_fetch_row($haku);
$osi = $tulos2[0];
$pohja = "http://193.166.152.141";
$yhd = $pohja . $tulo;
$selain1 = stristr($_SERVER['HTTP_USER_AGENT'], 'SymbianOS');
$selain = substr($selain1, 0, 9);
$vselain = "SymbianOS";


if($oso == $osi) {
if($selain == $vselain) {
echo "<br />";
$koti = "/home/";
$loppu = "/listat/pls_lista.txt";
$paikannus = $koti . $usernimi . $loppu;
$murmur = file($paikannus);
foreach ($murmur as $line_num => $murut){
$murri = substr($murut, 0, -6);
echo "<a href=http://193.166.152.141/~$usernimi/$murri.mur>$murri</a><br
/>";
}
echo $selain;
}
elseif($selain != $vselain) {
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title><? echo $usernimi ?>n Mstream</title>
<link href="../nappula.css" rel="stylesheet" type="text/css" />
</head>

<body>
```

LIITE 2

```
<table width="90%" height="100%" border="1" cellpadding="2" cellspacing="5">
  <tr>
    <th width="20%" height="251" bgcolor="#33CCFF" scope="row"><p><img src="http://193.166.152.141/m2.bmp"></p>
                  Etusivu:
                  <form action="" method="post">
                <input type="hidden" name="kotoisa" value="kuin kotona" />
                <input type="submit" class="btn" value="Alkuun"  name="ekat"/>
                </form>


                          Ohjeita:
                         <form action="" method="post">
                <input type="hidden" name="helppari" value="apuva" />
                <input type="submit" class="btn" value="Ohjeistukset" name="ekat"/>
                </form>

                Pleikkarin luonti:
                <form action="" method="post">
                <input type="hidden" name="luopls" value="Luo_PLS" />
                <input type="submit" class="btn" value="Luo PLS" name="ekat"/>
                </form>
                <p>MP3:n hallinta:</p>
                <form id="form1" name="form1" method="post" action="">
                  <label>
                  <input type="hidden" name="tokat" value="tokat" />
                  <input name="mp3" type="submit" class="btn" value="Siirry" />
                    </label>
                  </form>
                <p>
                  Pleikkareiden tarkastelu: </p>
                <form action="" method="post">
                <input type="hidden" name="tsekpls" value="tsek_PLS" />
                <input type="submit" class="btn" value="Katso PLS" name="tutki"/>
                </form>

<form action="" method="post">
<input type="hidden" name="ulos" value="LOGOUT" /><br />
<input type="submit" class="btn" value="LOGOUT" name="ekat" />
</form>


<br /> <?
echo $_SERVER['HTTP_USER_AGENT'];
?>

                  </p></th>
```

LIITE 2

```
   <td width="70%" bgcolor="#FFFFCC"><p> </p>
 <?

if(isset($_POST['ekat'])) {
if(isset($_POST['ulos'])) {
$query3 = "UPDATE user SET osoite = 'NOT LOGGED IN' where kayttaja =
'$usernimi'";
$tarkistetaan3 = mysql_query($query3) or die('Haussa häikkää');
header("Location: http://193.166.152.141");
}
elseif(isset($_POST['luopls'])) {
?>
<?
$bskripti="sudo /bin/biisit " . $usernimi;
exec("$bskripti");
$lines=file('/home/' . $usernimi . '/listat/biisit.txt');
echo "Kappaleita yhteensä:<b> ";
echo count($lines);
echo "</b><br />";
echo "<br />";
?>
<form action="" method="post">
Valitse seuraavista kappaleet listasta :<br />
<?
foreach ($lines as $line_num => $line) {

echo "<input type=checkbox name=biitit[] value=$line>$line</input><br />";
}
?>
<br />

Anna soittolistalle nimi:<input type="text" name="uuspls" size="15" />
<br />
<input type="submit" name="tokat" value="Luo lista" />

</form>

<?

}

elseif(isset($_POST['kotoisa'])) {
?>
taas kotona
<?
}
elseif(isset($_POST['helppari'])) {
?>
Ohjeita
<?
```

```
}
}
elseif(isset($_POST['tutki']) || isset($_POST['erasoi'])) {
if (isset($_POST['erasoi']))
{
$pls_skripti="sudo /bin/testi " . $usernimi;
$listanimi = substr($_POST['valittu_lista'], 0, -5);
$poistapls="sudo /bin/plspoisto " . $usernimi . " " . $listanimi;
exec("$poistapls");
exec("$pls_skripti");

}

# Skripti, joka listaa käyttäjän playlistit

$pls_skripti="sudo /bin/testi " . $usernimi;
exec("$pls_skripti");


# Alustetaan kyseinen skriptin tuottama tiedosto
# fopen funktiota varten
$sijoituu1 ="/home/";
$sijoituu2 ="/listat/";
$pls_tiedosto="pls_lista.txt";
if(file_exists($sijoituu1 . $usernimi . $sijoituu2 . $pls_tiedosto)) {
$fp1=fopen($sijoituu1 . $usernimi . $sijoituu2 . $pls_tiedosto,'r');
$sisalto=fread($fp1,filesize($sijoituu1 . $usernimi . $sijoituu2 . $pls_tiedosto));
fclose($fp1);

# laskuri, joka laskee rivien määrä, joka kertoo
# samalla playlistien määrän

$pls_laskuri=count($sisalto);

$riveja = count(file($sijoituu1 . $usernimi . $sijoituu2 . $pls_tiedosto));


# Tämä määritelmä on sitä varten, että saa luettua tekstitiedostosta
# rivi kerrallaan radioboxeja varten.

$radioboxi = file($sijoituu1 . $usernimi . $sijoituu2 . $pls_tiedosto);
$rivi_counteri = count($radioboxi);

echo "<h2>Playlist</h2><p>";
echo "Playlistejä on $riveja<br>";
echo "Alla on lueteltu, mitä playlistejä sinulla on. <br>";
echo "Valitse playlist, jonka sisältöä haluat katsella.<p>";

# Itse playlist listauksen sisällön tulostus
?>
```

LIITE 2

```
<form action="" method="post">
<?

for ($gg=0; $gg < $riveja; $gg++){
$listanimi = substr($radioboxi[$gg], 0, -6);
?>
<input type="hidden" name="tutki" />
<input type="radio" name="valittu_lista" value="<? echo $radioboxi[$gg] ?>"
/><? echo $listanimi ?><br>
<?
}
echo "<input type=submit name=lomakee value=tarkastele>";
echo "<input type=submit name=muokkaa value=Muokkaus>";
echo "<input type=submit name=erasoi value=Poista><br>";
echo "</form>";
if (isset($_POST['erasoi']))
{
$listanimi = substr($_POST['valittu_lista'], 0, -5);
echo "Poistit listan: " . $listanimi;
}
elseif (isset($_POST['muokkaa']))
{
$listanimi = substr($_POST['valittu_lista'], 0, -5);
$sijoituu1 ="/home/";
$sijoituu2 ="/listat/";
$sijoituu3 ="/public_html/";
$listaval = $_POST['valittu_lista'];
$kaikkikpl = "biisit.txt";
$kipaleet = file($sijoituu1 . $usernimi . $sijoituu2 . $kaikkikpl);
$listasis = file($sijoituu1 . $usernimi . $sijoituu3 . $listaval);
?>
<form action="" method="POST">
<?
echo $listanimi . "<br>";
foreach ($kipaleet as $line_num => $kpl) {

if(in_array($kpl, $listasis)){
echo "<input type=checkbox name=biitit[] value=$kpl
checked><b>$kpl</b></input><br />";

}
elseif(!in_array($kpl, $listasis)){
echo "<input type=checkbox name=biitit[] value=$kpl>$kpl</input><br />";

}

}
?>
<input type="hidden" name="tokat" />
<input type="hidden" name="uuspls" value="<? echo $listanimi ?>" />
```

```
<input type="submit" name="muokattulista" value="Tee muutokset" />
</form>
<?
}
elseif (isset($_POST['lomakee']))
{

# halutun playlistin tuloste
$listanimi = substr($_POST['valittu_lista'], 0, -5);
echo "Playlistin <a href=http://193.166.152.141/~$usernimi/$listanimi.pls> $lis-
tanimi </a> sisältö:<br />";
$tiedosto99= $_POST['valittu_lista'];
$fp=fopen($tiedosto99,'r');
$sisalto99=fread($fp,filesize($tiedosto99));
fclose($fp);
$sisalto99 = "<p>".$sisalto99."</p>";
$sisalto99 = str_replace(chr(10),"<br>",$sisalto99);
echo $sisalto99;

}
}
else {
echo "Playlisteja ei voitu hakea! Syynä on luultavasti se, että niitä ei ole yhtään";
}
}

if(isset($_POST['tokat'])) {
if(isset($_POST['uuspls']) || isset($_POST['muokattulista']))
{

$valitut="" . join(" ", $_POST['biitit']) . "\n";

$laskuri=count($_POST['biitit']);
echo "Valitsit $laskuri kappaletta, listalle:" . $_POST['uuspls'];
echo "<br>";
echo "Kappaleet ovat $valitut \n";
echo "<br>";

$lista = $_POST['uuspls'];
$plsluontiskr="sudo /bin/pls " . $usernimi . " " . $lista . " " . $valitut;

exec("$plsluontiskr");
}
elseif(isset($_POST['mp3']))
{
if(isset($_POST['poista']))
{
                $poistelu = "sudo tuhomp3 " . $usernimi . " " .
$_POST['poistettava'];
                exec("$poistelu");
```

```
}

$bskripti="sudo /bin/biisit " . $usernimi;
exec("$bskripti");


?>
Siirrä kappaleet omasta hakemistosta käyttöön:
<form action="" method="post">
<input name="movee" type="submit" value="Siirr&auml; kappaleet" /><br /><br
/>
<b> Tai tuhoa allovevista jokin:</b><br />
<?
$omatkipaleet = file('/home/' . $usernimi . '/listat/biisit.txt');
foreach ($omatkipaleet as $line_num => $hitti){
echo "<input name=poistettava type=radio value=$hitti /> $hitti <br />";
}
?>
<input name="tokat" type="hidden" value="tokat" />
<input name="mp3" type="hidden" value="mp3" />
<input name="poista" type="submit" value="Tuhoa" />
</form><br /><br />
<?
if(isset($_POST['movee']))
                {
                $siirtely = "sudo movemp3 " . $usernimi;

                exec("$siirtely");
                echo "<br>kappaleet on siirretty";
                }
if(isset($_POST['poista']))
                {
                echo "Poistit kappaleen: " . $_POST['poistettava'];
                }
}
}
?>
                </td>
  </tr>
</table>
</body>
</html>
<?
}
}
else {
header("Location: http://193.166.152.141/sivu2.php");

}
?>
```