LAPPEENRANNAN TEKNILLINEN YLIOPISTO
Tietotekniikan osasto
Tietoliikennetekniikan laitos

**Harri Hämäläinen**

# WWW-pohjainen opintojensuunnittelutyökalu

Diplomityön aihe on hyväksytty Lappeenrannan teknillisen yliopiston tietotekniikan osaston osastoneuvoston kokouksessa 20.4.2005.

Työn tarkastajina toimivat Prof. Jari Porras ja dosentti Jouni Ikonen sekä ohjaajana Prof. Jari Porras.

Lappeenrannassa, 25.11.2005

Harri Hämäläinen

Korpimetsänkatu 6-8 C 2

53850 Lappeenranta

+358 (0)50 522 4778

## TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto

Harri Hämäläinen

**WWW-pohjainen opintojensuunnittelutyökalu**

Diplomityö

2005

74 sivua, 18 kuvaa, 2 taulukkoa ja 5 liitettä.

Tarkastajat: Professori Jari Porras ja dosentti Jouni Ikonen.

Hakusanat: henkilökohtainen opintojensuunnittelu, WWW-ohjelmointi
Keywords: personal study plan, WWW programming

Opintojenohjauksen tarve on kasvanut viime vuosien aikana suomalaisissa yliopistoissa. Kasvaneet valinnanmahdollisuudet ovat monimutkaistaneet tutkintorakenteita ja tätä kautta opiskelijoiden saaman henkilökohtaisen opintojensuunnittelun tarve on korostunut. Yliopistoilla on kuitenkin käytössään rajalliset resurssit opintojen ohjaukseen, joten ohjausta pyritään tehostamaan mutta samalla säilyttämään palvelun laatu. Tarkempi ja yksityiskohtaisempi ohjeistaminen sekä opintosuunnitelmien mallipohjat ovat auttaneet opiskelijoiden tehtävää. Samalla opiskelijoilta on alettu vaatia henkilökohtaisen suunnitelman laatiminen jo opintojen alkuvaiheessa.

Tässä diplomityössä on toteutettu tutkintorakenteeseen perustuva WWW-pohjainen opintojensuunnittelutyökalu. Järjestelmän toiminta perustuu relaatiotietokantaan mallinnettuun tutkintorakennetietoon ja sen hyödyntämiseen. Opiskelijat voivat käyttää tätä tietoa laatiessaan opintosuunnitelmiaan ja opintosuunnittelijat tarkastaessaan opiskelijoiden suunnitelmia. Järjestelmän avulla opintojensuunnittelua ja sen ohjaamista voidaan tehostaa ja samalla opiskelijoille voidaan tarjota keskitetty tietovarasto opintosuunnitelmansa säilyttämiseen ja ylläpitoon.

Työn tuloksena toteutettu järjestelmä julkaistiin Lappeenrannan teknillisen yliopiston tietotekniikan osaston opiskelijoiden käyttöön keväällä 2005. Kerätyn saadun palautteen perusteella voidaan sanoa, että tämäntyyppiselle palvelulle osana yliopiston tarjoamaa opintojen ohjausta on olemassa oikea tarve.

## ABSTRACT

Lappeenranta University of Technology
Department of Information Technology

Harri Hämäläinen

**WWW-based tool for personal study planning**

Master's Thesis

2005

74 pages, 18 figures, 2 tables and 5 appendices.

Supervisors: Professor Jari Porras and Adjunct Professor Jouni Ikonen.

Keywords: personal study plan, WWW programming

The need for guidance of studies has been rising in Finnish universities during the last years. Growing rights of options has complicated the structures for degrees creating an increasing need for personal guidance. Since universities have limited resources in their use the guidance is tried to make more efficient meanwhile maintaining the quality of service. More precise and detailed guidance and examples for curriculas have helped the students so far. Meanwhile students are required to complete a personal study plan in the beginning of their studies.

A WWW-based tool for curriculum-based personal study planning has been created in this thesis. The functionality of the system is based on the use of curriculas that are modelled in the relational database. Students can utilize the ready-made structures while creating their study plans and tutors can use this information while checking the plans. Using this system study planning can be made more efficient and meanwhile offer a centralized database to storing and maintaining their plans.

The system was published for the use of the students in Department of Information Technology at Lappeenranta University of Technology during the spring 2005. The gained feedback has shown that there is a real need for this kind of a service as a part of university's student services.

# Alkusanat

Tämä työ on tehty Lappeenrannan teknillisen yliopiston tietotekniikan laitoksella. Työn rahoitus on saatu opetusministeriön rahoittamasta W5W-hankkeesta.

Haluan kiittää professori Jari Porrasta ja dosentti Jouni Ikosta mielenkiintoisesta aiheesta sekä työn toteuttamisen ja kirjoittamisen aikana saamastani ohjauksesta. Teidän alaisena on ollut miellyttävää työskennellä niin tämän kuin aikaisempienkin projektien puitteissa.

Haluan myös kiittää tietoliikennetekniikan laitoksen muuta henkilökuntaa mielenkiintoisesta ajasta ja ympäristöstä vuosien ajalta. Kimmo, Tomi, Jani, Jussi, Kari ja muut; jokainen teistä on tuonut oman mausteensa mielestäni viihtyisään työilmapiiriin.

Erityisesti haluan kiittää vanhempiani, Hilkkaa ja Timoa, jotka ovat vuosien saatossa yrittäneet potkia minua eteenpäin. Joskus neuvot ovat menneet perille, toisinaan ei.

Lappeenrannassa, 25. marraskuuta 2005.

Harri Hämäläinen

# Sisällysluettelo

# Lyhenneluettelo

| | |
|---|---|
| ANSI | American National Standards Institute. Standardointiorganisaatio. |
| ASCII | American Standard Code for Information Interchange. |
| BSD | Berkeley Software Distribution License. Ohjelmistolisenssi. |
| CPL | Common Public Licence. Ohjelmistolisenssi. |
| CSRF | Cross-Site Request Forgeries. WWW-sovelluksia vastaan käytettävä hyökkäystapa. |
| CSS | Cascading Style Sheets. HTML-sivuilla käytettävä tyylimäärittely. |
| DOT | Graphviz-ohjelmiston graafien kuvaamisessa käytettävä kieli. |
| DTD | Document Type Definition. SGML ja XML-dokumenttien sisällönkuvausmäärittely. |
| ECTS | European Credit Transfer System. |
| eHOPS | Elektroninen HOPS. Opintosuunnitelman tekoon käytettävä sähköinen järjestelmä. |
| HOPS | Henkilökohtainen opintojen suunnittelu tai Henkilökohtainen opintosuunnitelma. |
| HTML | Hypertext Markup Language. WWW-sivujen asetteluun käytettävä kuvauskieli. |
| HTTP | Hypertext Transfer Protocol. Internetin siirtoprotokolla. |
| ID | Identification. Tunniste. |

| | |
|---|---|
| IDN | Internationalized Domain Names. Erikoismerkkejä sisältävät domain-osoitteet. |
| IETF | Internet Engineering Task Force. Internetin teknisistä määrityksistä vastaava tekijä. |
| IP | Internet Protocol. Tiedonsiirtoprotokolla. |
| ISO | International Organization for Standardization. Kansainvälinen standardointiorganisaatio. |
| JOOPAS | Joustava opinto-oikeus. |
| LAMP | Linux, Apache, MySQL, PHP. Avoimista sovelluksista muodostuva WWW-palvelunalusta. |
| LTY | Lappeenrannan teknillinen yliopisto. |
| LUT | Lappeenranta University of Technology. |
| MD5 | 128 bittinen salausalgoritmi. |
| NAT | Network Address Translation. Mahdollistaa eri IP-osoitteiden käytön sisä- ja ulkoverkossa. |
| ODBC | Open DataBase Connectivity. Tietokannan hallintarajapinta. |
| OVI | Ohjausta virtuaalisesti. Joensuun yliopiston koordinoima Suomen Virtuaaliyliopiston osahanke. |
| PHP | PHP: Hypertext Preprocessor. Skriptikieli. |
| PNG | Portable Network Graphics. Kuvaformaatti. |
| RADIUS | Remote Authentication Dial-In User Service. |
| RFC | Request For Comments. IETF:n julkaisema määrittely. |
| SGML | Standard Generalized Markup Language. Metakieli dokumenttien rakenteen merkitsemiseen. |
| SOAP | Simple Object Access Protocol. Protokolla, joka mahdollistaa eri alustoilla toimivien sovellusten välisen tiedonsiirron. |

| | |
|---|---|
| SQL | Structured Query Language. Tietokannan käsittelyyn käytettävä kyselykieli. |
| SSL | Secure Socket Layer. Netscapen kehittämä protokolla tiedon salattuun lähettämiseen. |
| URI | Uniform Resource Identifier. Internetin resurssiin viittaava merkkijono. |
| URL | Uniform Resource Locators. Internetin resurssiin viittaava merkkijono, URI:n tyyppi. |
| W3C | World Wide Web Consortium. Kansainvälinen konsortio Web standardien kehittämiseen. |
| W5W | Walmiiksi Wiidessä Wuodessa. |
| WebTUTOR | Web sTUdy Track ORganizer. Diplomityössä toteutettu sähköinen opintojensuunnittelutyökalu. |
| WSDL | Web Services Description Language. XML-muotoinen palveluiden kuvauskieli. |
| WWW | World Wide Web. Hypertekstipohjainen hajautettu tietojärjestelmä. |
| XML | Extensible Markup Language. Metakieli dokumenttien rakenteen merkitsemiseen. |
| XSS | Cross-Site Scripting. WWW-sovelluksia vastaan käytettävä hyökkäystapa. |

# Luku 1

# Johdanto

Opintojen suunnittelun tarpeellisuuteen on viime vuosien aikana kiinnitetty entistä enemmän huomiota. Henkilökohtaisen opintosuunnitelman (HOPS) avulla pyritään hahmottamaan tutkinnon suorittamiseen vaadittavaa aikaa ja samalla saada opiskelijat asettamaan tavoitteita oman opintopolkunsa varrelle. Suunnitelman avulla yritetään jo hyvissä ajoin asettaa tavoitteita, joita kohti opiskelija pyrkii. Samalla pyritään kartoittamaan, minkätyyppisiä opintoja tutkintoon sisällytetään, mitkä opinnot opiskelijan on pakko suorittaa ja mitä opintoja hän voi vapaasti valita. Hyvissä ajoin laadittu suunnitelma voi auttaa opintojen kuormittavuuden tasapainottamisessa ja samalla pyritään etenemään opinnoissa loogisessa järjestyksessä, perusopinnoista kohti syventäviä opintoja. Huolella laadittu suunnitelma säästää kaikkien osapuolten työtä ja aikaa. Opiskelija ei epähuomiossa suorita tutkintonsa kannalta epäoleellisia opintoja, häntä ohjataan hankkimaan vaadittavat esitiedot ja näin pyritään säästämään samalla opetusta tarjoavan organisaation resursseja.

Omien opintojensa organisoinnin kannalta opiskelijan on järkevää laatia ensimmäinen HOPS mahdollisimman aikaisessa vaiheessa opintojaan. Nykyisin ensimmäinen versio opintosuunnitelmasta laaditaan usein ensimmäisen opintovuoden aikana osana jonkin opintojakson suoritusta. Ensimmäisestä versiosta muodostuu lopulta alustava suunnitelma, joka muuttaa muotoaan opintojen edetessä. Se kuitenkin tarjoaa opiskelijalle valmiin kehyksen, jonka puitteissa niin lyhyellä kuin pidemmälläkin tähtäimellä pyritään etenemään. Suunnitelman avulla opiskelijalla on joka tapauksessa mahdollisuus hahmottaa seuraavassa vaiheessa, käytännössä seuraavana lukuvuotena, ajankohtaisiksi tulevat vaatimukset. Toiveena on, että opiskelijat päivittävät suunnitelmaansa säännöllisesti ja myös tarkastelevat omaa etenemistään asetettuun aikatauluun verrattuna. Organisaation lähtökohdista katsoen opintojen järkevällä

suunnittelulla on mahdollisuus saavuttaa parempia tuloksia sekä lyhyempiä valmistumisaikoja.

Opintojen ohjaaminen vaatii nykyisellään henkilöstöresursseja ja siihen sisältyy monia eri vaiheita ennen opiskelijan valmistumista. Syksyllä 2005 käyttöön otettu kaksiportainen tutkintojärjestelmä vaatii entistä enemmän opiskelijoiden ohjaamiseen tarvittavaa aikaa. Kokonaisuudessaan siirtymävaihe kestää tekniikan alalla aina vuoteen 2010 asti. Näiden vuosien aikana osa opiskelijoista etenee vanhan tutkintorakenteen mukaan, kun samanaikaisesti uudet opiskelijat etenevät uuden kaksiportaisen tutkinnon asettamien vaatimusten pohjalta. Tutkintorakenteen ja opintojenohjauksen osittaisella virtualisoimisella pyritään ohjaamaan ja helpottamaan opiskelijan valintoja ja samalla vähentämään ohjaajien työtaakkaa.

Tässä diplomityössä toteutettiin WWW-pohjainen järjestelmä, jonka avulla voidaan luoda ja ylläpitää sähköistä opintosuunnitelmaa. Työ toteutettiin osana opetusministeriön rahoittamaa W5W-hanketta[1], johon Lappeenrannan teknillinen yliopisto (LTY) osallistuu vuosien 2004-2006 aikana. Työssä toteutettava järjestelmä on suunniteltu vastaamaan erityisesti tietotekniikan osaston tarpeita, mutta se on tarvittaessa myös muiden osastojen käytettävissä. Työ on toteutettu avoimeen lähdekoodiin perustuvana ja sovelluksen lähdekoodi on vapaasti saatavilla. Siten se on myös muiden organisaatioiden vapaasti käytettävissä.

## 1.1 Työn tavoite

On erittäin todennäköistä, että sähköisen opintojensuunnittelujärjestelmän tarjoaminen opiskelijoiden käyttöön tulee lähitulevaisuudessa osaksi yliopistojen tarjoamaa opintojenohjausta. Tässä diplomityössä perehdytään sähköiseen opintojenohjaukseen ja toteutetaan sovellus, jota voi hyödyntää opintojen suunnittelun välineenä opintojen alusta aina niiden loppuun asti. Työkalun avulla pyritään opiskelijan kannalta helpottamaan HOPSin luomista ja vähentämään opintojen suunnitteluun ja ohjaukseen tarvittavaa työmäärää. Tähän tavoitteeseen pyritään automatisoimalla opintojen ohjaukseen liittyvät mekaaniset vaiheet. Kaikkia henkilökohtaisen ohjauksen etuja ei luonnollisesti pystytä eikä pyritä saavuttamaan, mutta tavoitteena on tarjota opiskelijoille konkreettinen väline, jota apuna käyttäen tutkintorakenteen vaatimuksiin perustuvan suunnitelman luominen ja ylläpidettävyys helpottuvat huomattavasti nykyisestä.

[1] `http://www.w5w.fi/`

Työssä toteutettava järjestelmä, *WebTUTOR*, on WWW-pohjainen sovellus sähköisen opintosuunnitelman, eHOPSin, ylläpitoon. Sen lähtökohtana on, että opiskelijan laatima opintosuunnitelma perustuu jo valmiiksi opinto-oppaassa tutkinnolle asetettuihin vaatimuksiin ja noudattaa sitä rakenteeltaan. Järjestelmää ei ole tarkoitettu ainoastaan opintojen suunnitteluun ja aikatauluttamiseen niiden alkuvaiheessa, vaan sitä voidaan päivittää ja hyödyntää myös opintojen edetessä aina tutkintorakenteen lopulliseen hyväksyttämiseen asti.

Nykyisin henkilökohtainen opintosuunnitelma luodaan usein tekstinkäsittely- tai taulukkolaskentaohjelmalla, jonka jälkeen se tulostetaan paperille. Tällöin opintosuunnitelman päivittäminen ja seuraaminen tapahtuu harvakseltaan. Toteutettava järjestelmä toimii keskitettynä tietovarastona ja työkaluna sekä opiskelijoille että opintojen ohjaajille. Järjestelmän tarkoituksena ei ole tuottaa opiskelijaa sitovaa tai hänen opintojaan kontrolloivaa suunnitelmaa, vaan tarjota muokattava sähköinen dokumentti ja vaihtoehtoinen työkalu opintojen suunnitteluun.

Opintojen ohjaajauksen kannalta työkalun avulla pystytään tarjoamaan opiskelijoille entistä parempaa ohjausta. Oikein hyödynnettynä ohjausta voidaan tehostaa, kun osittain mekaaniset vaiheet voidaan myös ohjaajien kannalta muuttaa mahdollisimman läpinäkyviksi. Jo kertaalleen tehtyä ja hyväksyttyä suunnitelman osaa ei tarvitse uudelleen tarkastaa.

Vaikka toteutettava työkalu voidaan helposti mieltää pelkäksi opintojensuunnitteluvälineeksi, sen tuottamia raportteja voidaan käyttää myös muihin tarkoituksiin. Yhtälailla järjestelmän avulla voidaan laatia valmistumissuunnitelma, joka on opintosuunnitelmasta poiketen virallinen asiakirja, jonka avulla opiskelija voi saada tai menettää oikeuksia. Samainen dokumentaatio on käytettävissä lopullista tutkintorakennetta hyväksytettäessä.

Järjestelmä toteutetaan rakenteeltaan sellaiseksi, että se on hyödynnettävissä eri laitoksilla riippumatta niiden tutkintorakenteiden sisällöstä tai muodosta. Tämä mahdollistetaan niin, että tutkintorakenteet sekä kurssien, opintokokonaisuuksien ja tutkintorakenteiden väliset keskinäiset suhteet mallinnetaan tietokannassa, ei niinkään logiikkana itse sovelluksen lähdekoodissa. Tämän ansiosta järjestelmä säilyy toimintakykyisenä myös tutkinnossa tapahtuvien rakenteellisten muutosten jälkeen.

## 1.2 Työn rajaus

Työtä rajattaessa jaettiin eri osa-alueet omiin osiinsa. Näistä järjestelmän toiminnan rajaus muodosti tietyn perustan, joka heijastui myös muihin osiin. Olemassaolevat järjestelmät sekä niiden kanssa mahdolliset rajapinnat sekä sovelluksen käyttöliittymä voitiin rajata olemassaolevien tarpeiden perusteella.

Kukin opiskelija voi luoda itselleen yhden opintosuunnitelman. Järjestelmän suunnittelussa kuitenkin huomioidaan mahdollisuus useamman opintosuunnitelman samanaikaiseen ylläpitoon, mutta se nähtiin toistaiseksi tarpeettomana. Opiskelijan itselleen luomaan opintosuunnitelmaan tulee olla pääsy ainoastaan hänellä itsellään. Siinä vaiheessa, kun opiskelija päättää julkaista suunnitelmansa, on siihen pääsy lisäksi opettajatuutorilla tai opintosihteerillä.

Tällä hetkellä on olemassa erilaisia järjestelmiä, jotka sisältävät tietoa kursseista, tutkintorakenteesta ja opintosuorituksista. Ne eivät pääsääntöisesti tarjoa julkista rajapintaa ulkopuolisille palveluille, jonka avulla tiedon siirtäminen eri järjestelmien kesken voisi tapahtua automaattisesti ja olisi tarkoin määriteltyä. Niinpä toteutusvaiheessa tulee ottaa huomioon mahdollisesti tulevaisuudessa tapahtuva kehitys tällä rintamalla ja pyrittävä mahdollistamaan jo tässä vaiheessa erilaisten rajapintojen mahdollisimman yksinkertainen käyttöönotto. Nykyisellään kurssien ja tutkintorakenteiden päivittämiseen tarvitaan kuitenkin manuaalista työtä.

Järjestelmän käyttöliittymää toteutettaessa tulee ottaa huomioon kolmen eri käyttäjäryhmän tarpeet. Opiskelijat käyttävät järjestelmää suunnitellessaan omia opintojaan, opettajatuutorit tai opintoneuvojat tarkastavat opintosuunnitelmat ja järjestelmän ylläpitäjä päivittää järjestelmän tietoja, kuten lisää vuosittain uuden tutkintorakenteen. Käyttöliittymiä suunnitellessa tulee pääpaino suunnata helppokäyttöisyyteen sekä toiminnan nopeaan omaksumiseen.

Tutkintorakenne- ja kurssitieto tallennetaan relaatiotietokantaan. Järjestelmä mahdollistaa eri vuosien tutkintorakenteiden osien yhdistämisen ja lopullisen kokonaisuuden muodostamisen erillisistä opintokokonaisuuksista. Järjestelmä tukee myös uutta kaksiportaista tutkintojärjestelmää sekä opintojen ECTS (European Credit Transfer System) pisteytystä. Järjestelmän toteutuksen ohessa tietokantaan mallinnetaan LTY:n tietotekniikan osaston tutkintorakenteet alkaen vuodesta 1999.

## 1.3 Työn rakenne

Työ jakautuu rakenteeltaan niin, että ensin tutustutaan henkilökohtaisen opintosuunnittelun nykytilaan ja sille asetettuihin vaatimuksiin. Samalla käydään läpi olemassaolevia sähköisiä HOPS-työkaluja. Tämän jälkeen tarkastellaan WWW-pohjaisen järjestelmän toteutuksessa huomioon otettavia teknisiä vaatimuksia kappaleessa 3. Tietoturvanäkökohtia on käsitelty melko laajasti, koska sen huomioiminen WWW-pohjaisten julkisten verkkopalveluiden toteutuksessa on varsin kriittinen osa. Samalla kartoitetaan muut sovellukseen liittyvät järjestelmät Lappeenrannan teknillisen yliopiston verkkoympäristössä.

Työn käytännön osuus aloitetaan määritelemällä järjestelmän eri käyttäjätyyppien roolit sekä tarkastelemalla lähemmin suunnittelussa huomioituja vaatimuksia. Teknistä osin keskitytään pääosin järjestelmän arkkitehtuuriin, tietokantaan ja käytettävyyteen. Kappaleessa 5 keskitytään yksityiskohtaisemmin järjestelmän toteuttamiseen liittyviin teknisiin ratkaisuihin ja toteutuksessa huomioituihin asioihin. Tämän lisäksi käsitellään myös järjestelmän yhteyteen toteutettuja rajapintoja.

Kappale 6 sisältää järjestelmän käyttöönottoon liittyvät vaiheet. Samalla tutustutaan käyttäjiltä saatuun palautteeseen ja heidän kokemuksiinsa. Kappaleessa esitetään myös, kuinka oma opintosuunnitelma luodaan lopullisen järjestelmän avulla.

Johtopäätöksissä esitetään yhteenveto sovelluksen kehityksen aikana tehdyistä huomioista. Tämän lisäksi pohditaan järjestelmän mahdollista roolia tulevaisuudessa osana yliopistojen palveluita.

# Luku 2

# Opintojensuunnittelun nykytila

Opintojen suunnitteluun ja niiden etenemiseen suunnitelmien mukaan on viime vuosien aikana kiinnitetty entistä enemmän huomiota. Samalla on ymmärretty erilaisten elektronisten opintojensuunnittelujärjestelmien tarjoamat mahdollisuudet ja niiden käyttöönottoon on pyritty mahdollisimman nopealla aikataululla. Yliopistot ovat pyrkineet kartoittamaan mahdollisuuksia erilaisten sähköisten ratkaisujen käyttöönottamiseen. Yksi osatekijä tähän varmasti on se, että opintosuunnitelman laatimisen puolesta on otettu kantaa jopa pääministeri Vanhasen hallituksen ohjelmassa [1]:

> *"Henkilökohtaiset opintosuunnitelmat otetaan käyttöön."*

Opiskelijoilla on tarve suunnitella opintojaan etukäteen ja pyrkiä jakamaan niitä tasaisesti opintopolkunsa varrelle. Opintojensuunnittelu on yksi osa suurempaa kokonaisuutta, kun opiskelijoiden valmistumisaikoja pyritään lyhentämään. Huolellinen opintojensuunnittelu on kaikkien osapuolten etu.

Yliopistoympäristössä opintojenohjauksen toimijat voidaan jakaa kolmelle eri tasolle. Ylimmällä tasolla on yliopiston taso, jossa määritellään opintojenohjauksen raamit sekä koordinoidaan annettavaa ohjausta organisaatiotasolla. Tästä alempi taso on yksikkö, joilla kullakin voi olla omaan toimintatapaan liittyviä yksilöllisiä sääntöjä. Opiskelijalle annettava ohjaus suoritetaan alimman tason toimesta, jolloin yksilöllinen ja henkilökohtaisesti annettava ohjaus kasvaa. Seuraava lista perustuu osittain Ossi Kokon valtakunnallisessa HOPS-seminaarissa esittämään jaotteluun ohjauksen eri toimijoiden välillä [2]:

- Yliopiston taso
  - Opiskelijapalvelut
  - Ura- ja rekrytointipalvelut
- Yksikön taso
  - Osasto, tiedekunta
  - Laitos
- Henkilötaso
  - Tuutorit
  - Tuutorivastaava tms.
  - Muu henkilökunta

Puhuttaessa teknisen sovelluksen välityksellä tapahtuvasta opintojenohjauksesta, asettuu se opiskelijan kannalta henkilökohtaisen ohjaamisen tasolle kunkin opiskelijan käyttäessä järjestelmää yksilöllisesti omiin tarpeisiinsa. Tutkintorakenteen tallentaminen ja ylläpito toteutettavan kaltaisessa järjestelmässä voi olla joko yksikkö- tai yliopistotasolla suoritettava tehtävä. Laajemmasta mittakaavasta katsottuna tällaisen järjestelmän ylläpito kuitenkin kuuluu yliopiston tasolla annettavaan opintojenohjaukseen.

Opintojensuunnittelun saralla on yhä edelleen avoimia kysymyksiä. Erilaiset opiskelijan tietosuojaan liittyvät asiat pohdituttavat ohjauksen parissa työskenteleviä henkilöitä. Kysymykset, kuten kenellä ja millä tarkkuudella ohjaajalla tulisi olla oikeus tutustua laadittuun suunnitelmaan, ovat vielä vailla lopullista vastausta. Pohdintaa aiheuttaa myös laaditun asiakirjan sitovuus; millä tavoin se sitoo toisaalta opiskelijaa, mutta myös mahdollisesti oppilaitosta, joka on suunnitelman hyväksynyt.

## 2.1 Opintojensuunnittelun erilaiset lähestymistavat

HOPSista ei tällä hetkellä ole olemassa sen merkitystä ja sisältöä tarkoin tai yksiselitteisesti rajaavaa määritelmää. Niinpä laadittu opintosuunnitelma voi sisältää erilaisia näkökulmia sen tekijästä tai ympäristön asettamista vaatimuksista riippuen. Opintosuunnitelmat ovat usein määritetty kahdella, toistensa sisällöstä poikkeavalla, käsitteellä: rajattu HOPS ja avoin HOPS.

*Rajattu HOPS* on konkreettinen suunnitelma opintojen etenemisestä, jonka ensimmäinen versio laaditaan jo opiskelun alkuvaiheessa. Se keskittyy pääosin suoritettavien opintojaksojen ja opintorakenteen valitsemiseen, suunnittelemiseen ja aikatauluttamiseen. Se sisältää opintojen etenemisen, opintojaksojen ja kokonaisuuksien valitsemisen sekä ajankäytön suunnitelman. Samalla voidaan kartoittaa mahdolliset opintosuoritusten väliset korvaavuudet. Tässä mallissa oppimisen ja suunnitelman nähdään etenevän samalla, kun saadaan kartutettua opintosuorituksia. [3] [4]

*Avoin HOPS* on ajattelutapa oppimiseen ja kehittymiseen, johon voi myös liittyä urasuunnittelun elementtejä. Sitä voi pitää salkkuna, joka täydentyy oppimisen edetessä ja yksilöllisten tavoitteiden tarkentuessa. Se sisältää oman alan pääotsikot ja niiden asiakokonaisuudet ja sen tavoitteena on seurata tieteellistä ja ammatillista kehittymistä. [3] [4]

Molemmilla HOPSin lähestymistavoilla on omat etunsa ja kannattajansa. Kussakin tapauksessa ratkaisuun vaikuttaa lopullinen kohderyhmä ja yliopiston ohjaukselleen asettamat tavoitteet ja vaatimukset. Rajattu HOPS tarjoaa sekä opiskelijoille että hallinnolle välineen, jonka avulla opintosuunnitelmaa voidaan konkreettisesti arvioida. Rajatusta HOPSista puhuttaessa vaarana voidaan nähdä, että se mekaanisena mallina saattaa päästää opiskelijan, kuten mahdollisesti myös ohjaajan, liian helpolla [5]. Avoin HOPS ei vastaavasti tarjoa niinkään selkeää mallia opintojen suunnitteluun. Työkalun, joka on suunnattu sekä opiskelijan että ohjaajan tarpeisiin, on luonnollista pyrkiä täyttämään pääosin rajatulle HOPSille asetetut vaatimukset.

## 2.2 Tutkintorakenneuudistuksen vaikutukset

Tutkintorakenneuudistukseen johtanut Bolognan prosessi sai alkunsa 1998, kun Saksan, Ranskan, Italian ja Iso-Britannian korkeakoulutuksesta vastaavat opetusministerit allekirjoittivat yhteisen julistuksen eurooppalaisten korkeakoulututkintojen järjestelmien harmonisoinnista, ns. Sorbonnen julistuksen. Julistuksen perimmäinen tavoite on synnyttää yhteinen eurooppalainen korkeakoulutusalue vuoteen 2010 mennessä, joka samalla lisää eurooppalaisen korkeakoulutuksen kilpailukykyä ja vetovoimaa muihin maanosiin verrattuna. Tähän pyritään lähinnä kuudella asetetulla tavoitteella [6]:

- Ymmärrettävät tutkintorakenteet
- Yhdenmukaiset tutkintorakenteet

- Opintojen mitoitusjärjestelmien käyttöönotto
- Liikkuvuuden lisääminen
- Laadunarvioinnin eurooppalainen ulottuvuus
- Korkeakoulutuksen eurooppalainen ulottuvuus

Suomessa Bolognan prosessin vaikutus konkretisoitui, kun yliopistotutkinnot uudistuivat syksyllä 2005. Yliopistojen opintoaikoja pyritään nopeuttamaan ja siirtymisaikaa toiselta asteelta korkeakoulutukseen lyhentämään opiskelijavalintajärjestelmää kehittämällä. Samalla otetaan käyttöön asteittain kaksiportainen tutkintorakenne, jonka vaatimukset mitoitetaan työmäärää vastaaviksi siten, että opinnot on entistä tehokkaammin mahdollista suorittaa tavoiteajassa. [1]

Kaksiportaisessa tutkintorakenteessa HOPSin laatimiseen kuuluvat tehtävät voidaan jakaa opintojen eri vaiheiden välillä. Aiemmankin tutkintomallin aikana ohjauksen eri vaiheet ovat noudattaneet samantyylistä linjaa siihen soveltunein osin. Jako perustuu osittain 13.4.2005 Kuopion valtakunnallisessa HOPS-seminaarissa ryhmätyön tuloksena esitettyihin tehtäviin eri vaiheissa olevien opiskelijoiden ohjaamiseen kesken:

- *1. vuosi*: Opiskelijan ohjaus akateemisiin opiskelutaitoihin ja tutkintorakenteeseen tutustuminen. Laaditaan hahmotelma kandidaattivaiheeseen asti, otetaan käyttöön kurssisuunnitelmat ja ajankäytön seuranta.
- *2. vuosi*: Suunnitelman seuranta ja tarkistaminen, sivuainevalinnat ja suuntautumisvaihtoehdot. Mahdollisuuksien rajoissa laaditaan alulle suunnitelma jo maisteriksi asti. Kandidaattivaiheen jälkeen voi tapahtua paljonkin muutoksia ja maisterivaiheeseen voidaan tulla monen eri väylän kautta.
- *3. vuosi*: Edelleen suunnitelmien seuranta ja tarkistaminen, kandidaattityön ohjaus ja kandidaattitutkinnon "laadun tarkkailu".
- *4. ja 5. vuosi*: Maisterivaiheen alussa korostuu uusi alku ja mahdolliset alan vaihdot ja täydentävät opinnot. Opintosuunnitelma muotoutuu valmistumissuunnitelmaksi.

## 2.3 Nykyiset opintojensuunnittelujärjestelmät

Ennen työkalun suunnittelun ja toteuttamisen aloittamista tarkasteltiin ja arvioitiin jo olemassaolevia opintojensuunnittelujärjestelmiä sekä niiden sisältämää toiminnallisuutta. Perehdymme tarkemmin seuraaviin opintojensuunnittelujärjestelmiin:

- Desmond (Helsingin yliopiston tietojenkäsittelytieteen laitos)
- eHopo (OVI-hanke, Suomen virtuaaliyliopisto)
- Korppi -järjestelmä (Jyväskylän yliopisto)
- Oodin tuleva eHOPS-järjestelmä

Kaikki esitetyt järjestelmät ovat WWW-pohjaisia työkaluja. Yhteinen tekijä näille on myös se, että niiden elinkaari on vasta alkuvaiheessaan. Pääsääntöisesti järjestelmien avulla luotava suunnitelma omaa rajatun HOPSin piirteet. Sinänsä tätä ei voi pitää yllätyksenä, sillä juuri teknisen lähestymistavan omaavan rajatun suunnitelman luominen teknisen sovelluksen avulla on luonnollisempaa, kuin humanistisemman ja enemmän henkilökohtaiseen käyttöön tarkoitetun avoimen HOPSin. Järjestelmiä tarkasteltiin opiskelijoiden ja heille suunnattujen toimintojen lähtökohdista.

### 2.3.1 Desmond

Desmond on toteutettu opiskelijoiden projektitöinä Helsingin yliopiston tietojenkäsittelytieteen laitoksen käyttöön. Se on työkalu opintojen suunnittelun avuksi, jolla voi laatia suunnitelman lukukausi kerrallaan tai koko opiskeluajaksi. Järjestelmän idea perustuu opintojaksojen valitsemiseen ja niiden aikatauluttamiseen, ei niinkään tutkintorakenteeseen perustuvan loogisen opintopolun suunnittelemiseen. Järjestelmä on toteutettu asteittain useassa peräkkäisessä projektissa niin, että uusi ryhmä on lisännyt järjestelmään uusia toimintoja. Desmondin avulla laadittu suunnitelma on lähestymistavaltaan rajattu HOPS.

Suunnitelman pohjana voidaan lisäksi hyödyntää opintorekisteriotetta, jonka tiedot siirretään järjestelmään yliopistolla käytössä olevan Oodin tietokannasta. Desmondia on käytetty esimerkkitapauksena Oodi-konsortion tuottamassa vertailussa eri tiedonsiirtorajapintojen välillä. Desmond -järjestelmää varten luodun rajapinnan avulla kyetään hakemaan tietyn opiskelijan suoritustiedot Oodi-järjestelmästä opiskelijanumeron perusteella. [7] [8] Tietojen lähdejärjestelmänä tässä tapauksessa toimii Oodi ja kohdejärjestelmänä Desmond. Rajapinnan avulla on tiedonsiirto

automatisoitu niin, että siirtäminen tapahtuu opiskelijan kannalta mahdollisimman näkymättömästi ja on yksinkertainen käyttää.

### 2.3.2 eHopo

eHopo on osa Joensuun yliopiston koordinoimaa virtuaaliyliopiston OVI-osahanketta[1]. Järjestelmän kehitystyö alkoi vuoden 2002 alussa ja pilotointi alkoi Joensuun yliopiston laitoksilla syyskuussa 2003. Järjestelmän koekäyttöön on osallistunut myös LTY:n sähkötekniikan osasto Jussi Salon ohjauksessa.

eHopon alusta on suunniteltu toimivaksi samanaikaisesti useiden eri yliopistojen kesken. Se on verkkopohjainen HOPS- ja portfoliotyökalu opintojen suunnitteluun, toteutukseen ja arviointiin. Käyttäjällä on myös mahdollisuus lisätä kurssivalikoimaansa opintojaksoja, joita järjestelmässä ei ole valmiiksi syötettynä.

Opintojen suunnitelmallisuus järjestelmässä perustuu opiskeltaviin kursseihin sekä niiden aikatauluttamiseen. Opiskelija voi määrittää suunnitelmassaan arvionsa opiskeluidensa pituudesta. Tämän jälkeen hän voi aikatauluttaa valitsemiensa kurssien suorituksen haluamalleen ajanjaksolle. HOPSiin lisätyt kurssit voivat olla eri tilassa, joko suoritettuna, meneillään olevana, tulevana tai hylättynä. HOPSin voi lähettää arvioitavaksi omalle ohjaajalleen. Käyttäjällä on myös mahdollisuus ylläpitää samanaikaisesti useita HOPSeja.

Järjestelmässä opiskelija voi itse määrittää valitsemansa opintojakson kuuluvaksi haluamaansa opintokokonaisuuteen, kuten aine- tai syventäviin opintoihin. Suunnitelmaan lisätyistä opintojaksoista saa halutessaan luettelon, jonka voi järjestää myös opintokokonaisuuksien mukaan, jolloin eri opintokokonaisuuksiin kuuluvat opinnot ovat selkeästi eroteltu toisistaan. Järjestelmän tietorakenne ei valmiiksi sisällä tietoa tutkintorakenteesta tai siihen sisältyvistä opintojaksoista, joten kurssien lisääminen opintosuunnitelmaan ja niiden siirtäminen haluttuihin opintoryhmiin on tehtävä käsin, eikä järjestelmä tarkasta tai ota kantaa kurssin sopivuudesta valittuun opintokokonaisuuteen.

Tarkastelluista järjestelmistä eHopo sisältää ainoana selkeästi osia myös avoimeen HOPSiin liittyen. eHopon opiskelijoille suunnattu toiminnallisuus vaikuttaa onnistuneelta, jossa toteutetun suunnitelman laatu ja sen yksityiskohtaisuus jää opiskelijan itsensä vastuulle. Ruudunkaappaus järjestelmän aikataulupohjaisesta näkymästä on esitetty kuvassa 2.1.

[1] `http://ovi.joensuu.fi/`

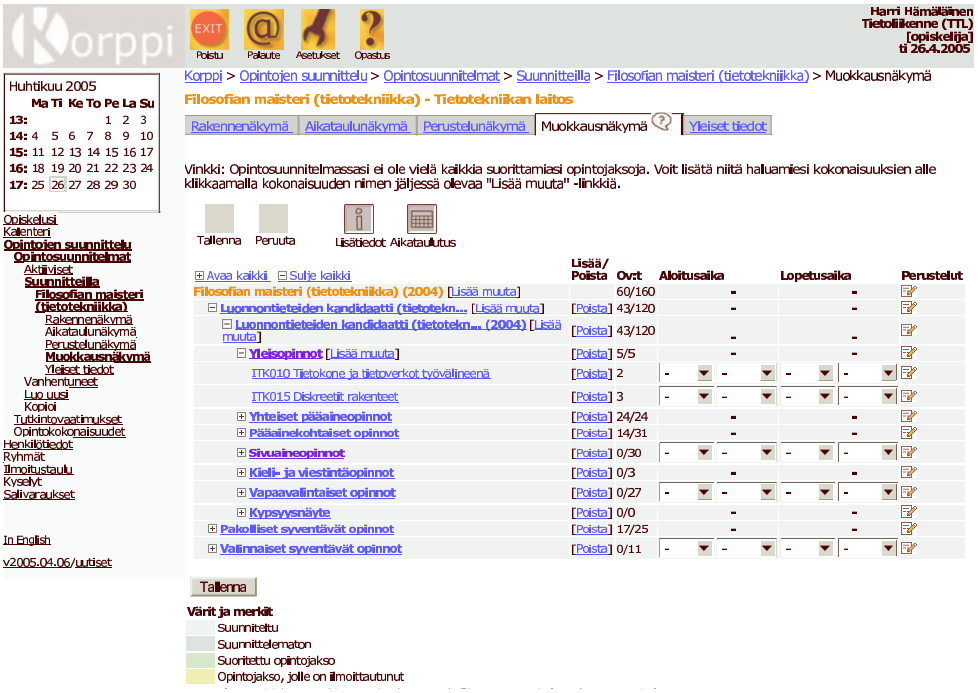

Kuva 2.1: Kuvankaappaus eHopon näkymästä. Opintosuunnitelma on esitetty aikataulutukseen perustuen.

### 2.3.3 Korppi -järjestelmä

Jyväskylän yliopistossa opintosuoritukset on tallennettuna Korppi-opintojärjestelmässä. Kuikka-projektissa[2] on toteutettu HOPS-järjestelmä, joka on integroitu Korpin yhteyteen. Alkuperäinen Korppi sisältää myös tutkintorakenne- ja kurssitietoa, joten sitä on pyritty hyödyntämään Kuikkaa toteutettaessa. Järjestelmä vastaa suurelta osin ajatustamme käytännöllisestä eHOPS-työkalusta. Integroidun järjestelmän pilotointi on aloitettu syksyn 2004 aikana.

Opiskelijat käyttävät sovellusta opintojensa rakenteen suunnitteluun, aikatauluttamiseen, perustelemiseen ja raportointiin. Opiskelijalla voi olla useita opintosuunnitelmia, mutta vain yksi kerrallaan niistä on aktiivinen tai virallinen. Opiskelija voi lähettää opintosuunnitelmansa suoraan hyväksyjälle ilman konsultointia opintojen ohjaajan kanssa. [9] Opintosuunnitelma voidaan hyväksyä, jonka jälkeen käyttäjällä ei ole enää mahdollisuutta sen muokkaamiseen. Hän voi kuitenkin kopioida sen uuden suunnitelman pohjaksi ja tarvittaessa hyväksyttää sen myöhemmin. Sovelluksen avulla luotava suunnitelma on malliltaan rajattu HOPS. Näkymä järjestelmän tutkintorakennenäkymästä on esitetty kuvassa 2.2.

[2] `http://sovellusprojektit.it.jyu.fi/kuikka/`




Kuva 2.2: Kuvankaappaus Korppi-järjestelmän näkymästä. Opintosuunnitelma on esitetty rakenteellisessa muodossa.

### 2.3.4 Oodi

Jo olemassa olevien järjestelmien lisäksi pyrimme kartoittamaan myös Oodiin kehitettävän järjestelmän toiminnallisuutta. Oodi[3] on yliopistojen opetus- ja opiskelutoimintojen tukemiseen tarkoitettu järjestelmä, jota käyttää ja ylläpitää kolmentoista suomalaisen yliopiston muodostama Oodi-konsortio yhdessä tietojärjestelmätoimittajien kanssa. On todennäköistä, että Oodi tulee olemaan monen yliopiston ratkaisu sähköisen opintosuunnitelman laatimiseen. Oodi on käytössä myös Lappeenrannan teknillisessä yliopistossa, mutta se ei toistaiseksi sisällä opintojen suunnitteluun tarkoitettua työkalua. Esiselvitys- ja määrittelytyö sen toteuttamisen suhteen on tehty.

Oodiin integroitavan eHOPS-työkalu projektin oli alunperin määrä olla valmiina syksyllä 2005. [10] Määrittelytyön perusteella toteutettava työkalu tulee sisältämään tiedekuntaan tai laitokseen sidottavia tutkintorakennepohjia. Opintosuunnitelmat pohjautuvat näihin järjestelmässä oleviin pohjiin. Opiskelijalla voi samanaikaisesti olla useampi
HOPS, joista kuitenkin yksi kerrallaan on ensisijainen. Hyväksyttyä suunnitelmaa ei voi enää muokata, mutta sen voi kopioida uuden suunnitelman pohjaksi. [11]

[3] `http://www.oodi.fi/`

# Luku 3

# Järjestelmän tekniset vaatimukset

Ennen järjestelmän suunnittelua pyrittiin kartoittamaan teknisille ratkaisuille asetettuja vaatimuksia ja niihin liittyvää teoriaa. Koska järjestelmä perustuu varsin mittavan tietokantarakenteen ja henkilötietokannan käyttöön, pyritään sen suunniteluun ja ylläpitoon liittyviin kysymyksiin etsimään vastaukset. Lisäksi perehdytään julkisten WWW-pohjaisten palveluiden tietoturvaan ja niiden mahdollisiin tietoturvariskeihin.

Erityisen tärkeäksi työssä koettiin riittävän hyvän tietoturvatason määrittäminen järjestelmään. Koska järjestelmän sisältämä tieto opiskelijoista on suurelta osin henkilökohtaista ei-julkista tietoa, ei se saa vuotaa ulkopuolisten käyttöön. Niinpä tässä kappaleessa eritetään erilaisia WWW-sovelluksia vastaan käytettäviä hyökkäysmetodeja.

## 3.1 Tietokannan asettamat vaatimukset

Ensimmäinen huomioonotettava seikka tietokantaa suunniteltaessa on pyrkiä mallintamaan kaikki tieto ja kokonaisuuksien välille muodostettavat relaatiot todellisuuden mukaisesti. Samalla tulee kuitenkin välttää redundanssia – samansisältöisen tiedon toistuvuutta – ja pyrkiä tallentamaan kukin tieto kantaan ainoastaan kertaalleen. Ylimääräisten elementtien käyttöä tulee myös pyrkiä välttämään ja relaatiot pyrittävä valitsemaan riittävän tarkasti. [12]

Tietokannan redundanssia pyritään hallitsemaan tietokannan normalisoinnin avulla. Se on vaiheittainen malli, jonka avulla relaatiotietokanta saadaan tukemaan tiedon ehjää tallennusta sekä mahdollisimman tehokasta tiedon saatavuutta. Normalisoin-

nin avulla tietokannasta saadaan poistettua päällekkäisyys eli samojen tietojen tallentaminen useaan eri paikkaan. Tietojen tallentaminen useaan paikkaan hankaloittaa tiedon ylläpidettävyyttä sekä kasvattaa tietokannan kokoa.

Koska palvelun tietokantaan tallennetaan opiskelijoiden itsestään antamaa henkilökohtaiseksi luokiteltavaa tietoa, on Suomen lain asettamat säädökset otettava huomioon. Henkilötietolaki (L 22.4.1999/523) toteuttaa yksityiselämän suojaa ja muita yksityisyyden suojaa turvaavia perusoikeuksia henkilötietoja käsiteltäessä sekä edistää hyvän tietojenkäsittelytavan kehittämistä ja noudattamista. Käytännössä tämä tarkoittaa rekisteriselosteen laatimista tallennettavasta tiedosta ja sen käytöstä. Henkilötietolain 10§ mukainen rekisteriseloste on liitteessä 1.

## 3.2 Käytettävyys

Myös WWW-pohjaisia sovelluksia toteutettaessa tärkeä huomioonotettava asia on sovelluksen käytettävyys. Huonolla käytettävyydellä voi pilata lähes kaikki hyvätkin sovellukset, joskaan hyvä käytettävyys ei yksin pelasta huonoa ohjelmistoa. Käytettävyyttä suunniteltaessa tulee kartoittaa kohderyhmän tarpeet, palvelun tyyppi ja sen tarjoama sisältö.

Käyttöliittymän arviointiparametrina käytettävyys on itse asiassa ulkoasun, miellyttävyyden ja tehokkuuden yhdessä muodostama arviointiparametri. Yhdessä sovelluksen hyödyllisyyden kanssa käytettävyys lopulta määrittää sen käyttökelpoisuuden ja tarpeellisuuden. Seuraavalla sivulla esitetyssä taulukossa 3.1 [13] on lueteltu käytettävyyden arvioinnissa käytettäviä mittausarvoja ja sitä, kuinka ne ilmenevät käyttäjälle:

Toteutettavan järjestelmän ollessa kyseessä käyttöliittymää toteutettaessa tulee huomioida sen yksinkertaisuus, helppokäyttöisyys sekä etenkin nopea omaksuttavuus. Usein näitä haettaessa joudutaan tinkimään järjestelmän käytön tehokkuudessa, sillä toimintojen on tällöin oltava loogisessa järjestyksessä eikä oikopolkuja eri asioiden toteuttamiseen ole syytä tarjota. Sovellus on kuitenkin rakenteeltaan riittävän yksinkertainen eikä se sisällä hierarkkisesti pitkiä toimintoketjuja. Koska todennäköisesti yksittäiset opiskelijat käyttävät järjestelmää harvakseltaan, erilaisten oikopolkujen toteuttaminen on melko hyödytöntä.

Taulukko 3.1: Nielsenin määrittämät käytettävyyden perusedellytykset.

| Edellytys | Määritelmä |
|---|---|
| Opittavuus | Kuinka helppo käyttäjien on suorittaa perustoiminnot ensimmäisellä käyttökerralla? |
| Tehokkuus | Kuinka nopeasti käyttäjät voivat suorittaa haluamansa toiminnot opittuaan mallin? |
| Muistettavuus | Kuinka nopeasti käyttäjät voivat muodostaa uudelleen taidon käyttää järjestelmää tauon jälkeen? |
| Virheet | Kuinka paljon käyttäjät tekevät virheitä, kuinka pahoja ne ovat ja kuinka helposti he voivat selvitä niistä? |
| Miellyttävyys | Kuinka miellyttävä mallia on käyttää? |

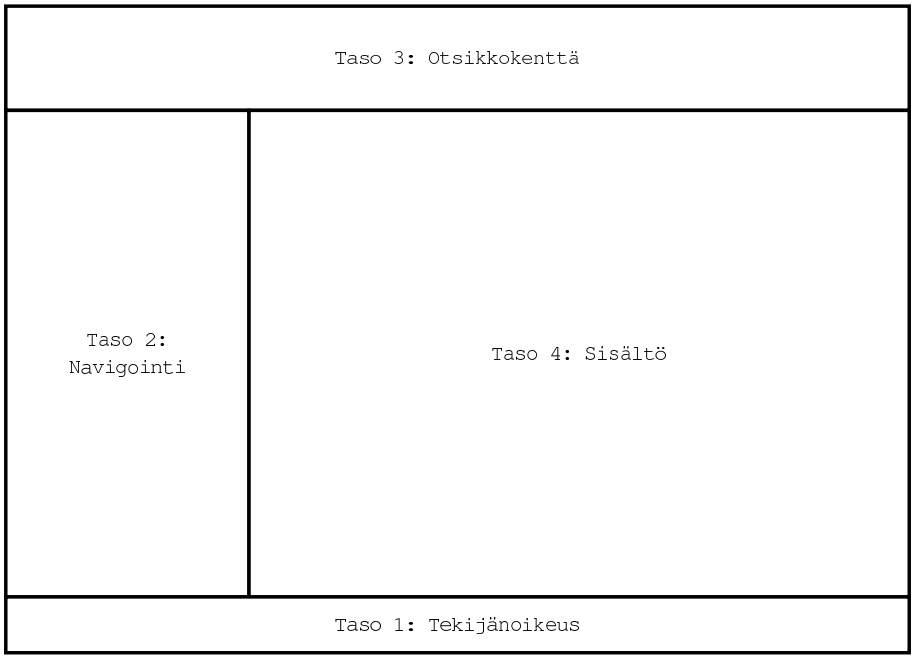
### 3.2.1 Käyttöliittymä

Käyttöliittymä on se osa, jonka käyttäjä sovelluksesta näkee. Käyttöliittymän suunnittelua edeltää kartoitus siitä, kuka ja minkälaiset käyttäjät tuotetta lopulta käyttävät ja kuinka sitä käytetään tukemaan tehtävien suoritusta. [14] Tässä työssä erilaiset käyttäjätyypit noudattavat aiemmin tehtyä määritelmää opiskelijoiden, ohjaajien ja ylläpitäjien kesken. Jokainen ryhmistä kuitenkin omaa hieman toisistaan eroavat käyttövaatimukset, tarpeet ja oikeudet järjestelmässä olevan tiedon käsittelyyn.

WWW-sivun käyttöliittymä voidaan sisällön ja toimintojen puolesta jakaa neljään erilliseen kategoriaan [15]:

- Taso 1. Räikeä informaatio, kuten mainokset, tekijänoikeus, koristelu jne.
- Taso 2. Hyödyllinen, mutta ei suoranaisesti aiheeseen liittyvä informaatio, kuten navigointi, hakemistorakenne jne.
- Taso 3. Sivun sisältöön nähden asiaankuuluva informaatio, mutta ilman huomattavaa tärkeyttä, kuten asiaan liittyvät otsikot, hakemisto jne.
- Taso 4. Sivun tärkein osa, johon kuuluu otsikot, sisältö jne.

Ihmiset ovat tottuneet WWW:tä käyttäessään tiettyihin perusrakennemalleihin, joihin suuri osa käytettävistä WWW-sivuista perustuu. Kuvassa 3.1 on esitetty tyypillinen navigointikeskeinen rakenne. Tässä mallissa sivuston toiminnan kannalta tärkeimmät osat pyritään sijoittamaan loogisesti tärkeimmille alueille.




Kuva 3.1: Navigointikeskeinen käyttöliittymämalli.

Ylin solu, joka aiemmin esitetyissä kategoriassa kuuluu tasolle 3, kertoo käyttäjälle, mitä palvelua hän käyttää. Tähän osaan sijoitetaan järjestelmän tunnistetieto, logo. Vasempaan soluun sisällytetään navigointitoiminnot, jotka kuuluvat kategoriaan 2. Suurimpaan esitetyistä soluista sijoitetaan sovelluksen toiminnan kannalta aktiivinen sisältö. Se on tärkein osa näkymää ja kuuluu kategoriaan 4. Alimmaiseen soluun voidaan sijoittaa mahdolliseen tekijänoikeuteen tai ylläpitoon viittaava teksti. Tämä osa kuuluu kategoriaan 1.

Sovelluksessa, jossa toimintojen suorituksella on selkeä järjestys, on toimivan navigoinnin toteuttaminen käytön miellyttävyyden kannalta ehdoton perusedellytys. Jacob Nielsenin mukaan [13] navigointiliittymän pitää pystyä vastaamaan kolmeen peruskysymykseen:

- Missä olen?
- Missä olen ollut?
- Minne voin mennä?

*"Missä olen?"*-kysymys voidaan esittää kahdella eri tasolla: WWW:iin tai sovellukseen liittyvänä. WWW-tasolla puhuttaessa sovellus voidaan personoida omalla ulkoasulla sekä omalla logolla. Tällöin käyttäjän on helppo käsittää, missä varsinaisesti on. Sovellustasolla puhuttaessa käyttäjän tulee ymmärtää, mitä hän on par'aikaa tekemässä. Tämä voidaan saada aikaan sivun otsikoinnilla sekä havainnollistamalla sijainti navigaatio-osassa. [13]

*"Missä olen ollut?"*-kysymykseen vastaamiseen voidaan vaikuttaa säilyttämällä hyperlinkkien värit alkuperäisen määrityksen mukaisena. Selaimet usein havainnollistavat jo vierailtuja linkkejä normaaleista linkeistä eroavilla väreillä. Myös selaimen "Takaisin"-nappi voi auttaa tähän kysymykseen vastaamisessa. [13]

*"Minne voin mennä?"*-kysymykseen vastauksen antaa navigointi ja muut sivulle sisällytetyt linkit. Käyttäjät ovat usein tottuneet käsittelemään alleviivattua tekstiä linkkinä. Linkit voidaan jakaa kolmeen erilliseen ryhmään: upotetut, rakenteelliset ja assosiatiiviset linkit. Upotetut linkit ovat tekstin seassa olevia linkkejä, rakenteellisten linkkien avulla navigoidaan sivulla ja assosiatiiviset linkit viittaavat sivun sisältöön liittyviin kohteisiin. [13]

### 3.2.2 Suorituskyky

Käyttäjän näkökulmasta WWW-sovelluksen suorituskyky riippuu eri tekijöistä. Tehtävän suorittaminen palvelimella vaatii aina toiminnon kompleksisuudesta riippuvan ajan. Käyttäjän kannalta yhtälailla merkittävä seikka on tiedon siirtoon vaadittava aika. Mitä vähemmän siirrettävää informaatiota on ja mitä nopeampi on käytettävä yhteys, sitä pienempiä sovelluksen vasteajat ovat.

Robert B. Miller on jo vuonna 1968 määritellyt rajat sovelluksen vasteajoille, jotka ovat tunnustettuja ja käyttökelpoisia vielä nykyisinkin. Miller on jakanut vasteajat kolmeen seuraavaan kategoriaan: [13] [16]

- Noin sekunnin kymmenesosa (0.1) on raja, jolloin käyttäjä kokee järjestelmän reagoivan välittömästi, jolloin tuloksen lisäksi ei tarvita erillistä palautetta.
- Noin sekunti (1.0) on raja, jolloin käyttäjä ei tunne ajatustoimintansa häiriintyväksi, vaikka huomaakin viiveen. Alle sekunnin kestävät toiminnot eivät kuitenkaan tarvitse erillistä huomautusta toiminnan aiheuttamasta viiveestä.
- Noin kymmenen sekuntia (10.0) on raja, jolloin käyttäjän huomion saa pidettyä dialogissa. Enemmän aikaa vaativat viiveet aiheuttavat sen, että käyttäjä

siirtyy tekemään muita toimintoja odottaessaan suorituksen valmistumista. Alle kymmenen sekuntia kestävät suoritukset käyttäjä kokee häiritsevinä, mutta säilyttää silti keskittymisensä.

WWW-sovelluksen käytöstä puhuttaessa tulee muistaa sen eroavaisuus perinteisiin lokaalisti ajettaviin sovelluksiin. Verkkopohjaisia sovelluksia käytettäessä vasteajat ovat säännöllisesti yli 0.1 sekuntia. Linkkiä seuratessaan käyttäjä kuitenkin havaitsee selaimestaan, että uutta sivua on alettu hakea, ja osaa odottaa sen latautuvan piakkoin. Niinpä huomiota on pyrittävä kiinnittämään siihen, ettei itse toiminnon suoritus ja tiedoston siirtoaika kasva liian suureksi. Mikäli näin käy, on käyttäjää varoitettava tästä etukäteen.

## 3.3 Tietoturva

Toteutettavan järjestelmän WWW-pohjaisuus ja sen saatavuus julkisesta verkosta, Internetistä, asettaa vaatimuksia ja haasteita palvelun tietoturvalle. Suuri osa palvelimelle tallennettavasta tiedosta on määritettävissä käyttäjien henkilökohtaisiksi tiedoiksi. Näihin tietoihin ei ulkopuolisten tule päästä käsiksi eikä tiedon luvaton tuhoaminen tule olla mahdollista.

WWW-pohjaisen järjestelmän tietoturva voidaan jakaa kolmeen erilliseen kategoriaan: verkkoarkkitehtuurin turvallisuuteen, palvelualustan turvallisuuteen ja itse sovelluksen turvallisuuteen. Ohjelmiston asennuksen yhteydessä tulee huomioida arkkitehtuuriin ja palvelualustaan liittyvät turvallisuusseikat. Tässä työssä keskitytään kuitenkin tarkastelemaan ja ratkaisemaan ongelma pääosin sovelluksen toteutuksen näkökulmasta.

WWW-pohjaisissa sovelluksissa asiakkaan ja palvelimen välillä välitetään tietoa molempiin suuntiin. Ellei tiedon sisältöä ja muotoa millään tapaa kontrolloida, aiheutetaan automaattisesti suuri tietoturvariski, jonka avulla uteliaat tai ilkeämieliset käyttäjät voivat joko hakea heille kuulumatonta tietoa tai muokata olemassa olevaa järjestelmää tai sen tietokantaa.

Tietoturvaa toteutettaessa tulee muistaa, että se on yhtä vahva kuin sen heikoin lenkki. Käytännössä tietoturvaan käytettäville resursseille ei ole ylärajaa ja tietojen suojaamiseen mahdollisesti käytettävät resurssit voivat kasvaa suhteettoman suuriksi. Riittävän huolellinen suunnittelu ja toteutus riittävät yleensä estämään tietovuodot, kun riskiä tarkastellaan sovelluksen näkökulmasta. Yleisenä ohjeena kannattaa muistaa, ettei tietojen suojaamiseen kuitenkaan kannata käyttää sitä enempää re-

sursseja mitä tietojen todellinen arvo on. Usein kuitenkaan tietojen arvoa ei voida konkreettisesti mitata, joten tietoturvaa toteutettaessa on pyrittävä kartoittamaan realistiset uhkat.

### 3.3.1 Palvelualustan turvallisuus

Palvelualustan järjestelmällinen ylläpito on suurin turvallisuuteen vaikuttava yksittäinen tekijä. Ylläpidon tulee ottaa tietoturvaseikat huomioon jo palvelinta asennettaessa ja konfiguroitaessa. Jatkossa tärkeää on riittävän tiheään tahtiin suoritettavat ohjelmistojen päivitykset. Palvelualustan ylläpitoon voidaan laskea kuuluvaksi ainakin käyttöjärjestelmän, WWW-palvelimen ja muiden sovellusten turvallisuudesta huolehtiminen ja niiden päivittäminen.

Mikäli palvelu asennetaan palvelimelle, jolle on asennettu muitakin erillisiä palveluita, tulee palvelimen sisältä tulevat hyökkäykset ottaa huomioon palvelinta konfiguroitaessa ja jo ohjelmiston suunnittelu- ja toteutusvaiheessa. Sovellustasolla tämä tarkoittaa, että esimerkiksi tietokantayhteyden muodostamiseen ja autentikointiin käytetyt tiedot ovat eritelty varsinaisesta sovelluksesta ja tallennettu omaan tiedostoon, johon ulkopuolisilla ei ole lukuoikeutta. Näihin kuuluu käyttäjätunnus, salasana, tietokantapalvelimen osoite ja tietokannan nimi. Palvelualustan turvallisuuteen ja tämän vaikutukseen taas liittyy, että kriittisiin resursseihin, kuten myös palvelimelle ylipäänsä, annetaan käyttöoikeus vain tarkoin rajoitetulle ja kontrolloidulle ryhmälle.

Osaksi palvelualustan turvallisuutta voidaan katsoa myös tietojen varmuuskopiointi. Palvelimen tiedot voivat tuhoutua esimerkiksi hyökkääjän toiminnasta tai laitteistoviasta aiheutuen. Tällöin on oltava mahdollisuus palata takaisin aiemmin olleeseen tilaan. Sovelluksen toteutuksessa ei varmuuskopiointia huomioitu, vaan se jätettiin osaksi palvelinalustan toimintaa.

### 3.3.2 Verkkoarkkitehtuurin ja tietoliikenteen turvallisuus

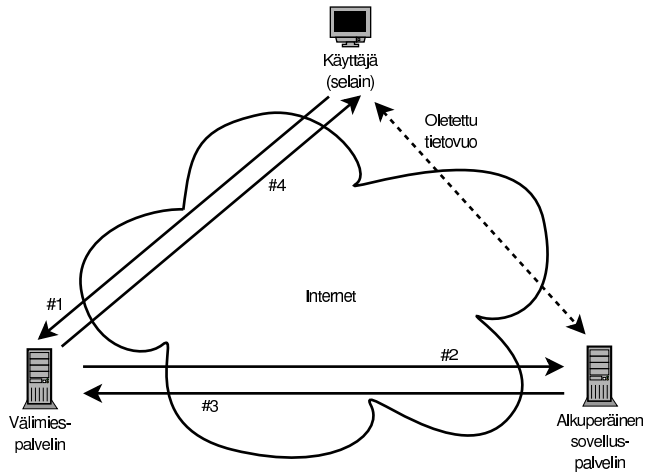
Verkkoarkkitehtuurin turvallisuuteen vaikuttaa palvelimen sijoittelu organisaation verkkoympäristössä. Palomuurin avulla kyetään estämään osa hyökkäyksistä. Palvelin saattaa olla eristetty julkisesta verkosta, jolloin sen resurssit ovat tarjolla ainoastaan sisäverkosta tuleville yhteyksille. Tässäkin tapauksessa tulee kuitenkin ottaa huomioon, että hyökkäys voi yhtälailla tulla sisäverkosta tai muuten luotettavaksi luokitellusta lähteestä.

Koska palvelun ja käyttäjän päätteen välillä tapahtuva tietoliikenne on henkilökohtaista tietoa, tulee liikennöinnin tapahtua salattuna. Käytännössä tämä on järjestetty niin, että kaikki verkkoliikenne asiakkaan ja palvelimen välillä kulkee salattuna. Tyypillinen tapa salata HTTP-pohjaisia yhteyksiä on käyttää SSL-protokollaa (Secure Socket Layer). Sen käytön tarkoituksena on taata yksityisyys ja luotettavuus kahden keskenään kommunikoivan ohjelmiston välillä. SSL-protokollan kehityksen tavoitteena on ollut luoda protokolla, joka on turvallinen, mutta samalla riittävän tehokas [17].

Verkkoliikennöintiin liittyvä uhka on myös välimieshyökkäys. Tällöin käyttäjän ja palvelimen väliseen tietoon pyritään pääsemään käsiksi esimerkiksi palvelimen kopion, reititystietojen tai nimipalvelimen tietojen väärentämisen avulla. Palvelimen kopion osoite voidaan välittää käyttäjälle mahdollisesti sähköpostitse tai tarjota linkki toisen palvelun kautta. Kaikki siirrettävä tieto kulkee välimiehen kautta. Vaikka selaimen osoitekentässä näkyvään osoitteeseen voidaan mahdollisesti vaikuttaa JavaScript:iä hyödyntäen, voi käyttäjä tässä tapauksessa varmentaa palveluntarjoajan SSL palvelinsertifikaatilla. Suuri osa verkkopalveluiden käyttäjistä kuitenkin hyväksyy sertifikaatit kiinnittämättä niihin ollenkaan huomiota. Lisäksi käyttäjiä voidaan harhauttaa myös harhaanjohtavien domain-nimien avulla.

Välimieshyökkäyksen toimintaperiaate on esitetty kuvassa 3.2. Käyttäjä olettaa asioivansa suoraan oikeellisen sovelluspalvelimen kanssa. Sitä vastoin kaikki liikenne kulkeekin välimiespalvelimen kautta, jolle käyttäjän yhteyspyyntö lähetetään vaiheessa 1. Kyseinen palvelin voi sisältää kopion alkuperäisestä sovelluksesta, tai voi tuottaa näkymän reaaliaikaisesti oikeellisen palvelun avulla. Sovellus voi tarvittaessa suorittaa määrätyt toiminnot automaattisesti vaiheiden 2-3 mukaisesti ja ohjata palautteen alkuperäiselle käyttäjälle vaiheessa 4. Samalla välimiehellä on kuitenkin mahdollisuus vaikuttaa kaikkeen asiakkaan ja palvelimen välillä siirrettyyn tietoon ja hyödyntää haltuunsa saamaansa tietoa tarvittaessa myöhemmin.

Välimieshyökkäystä voidaan tehostaa entisestään IDN-osoitehuijauksen avulla. IDN (Internationalized Domain Names) [18] mahdollistaa myös muiden, kun ASCII-merkkien käyttämisen URL-osoitteissa (Uniform Resource Locators). Monissa selaimissa tämän avulla kyetään vaikuttamaan osoiterivillä näkyvään osoitteeseen niin, että käyttäjän on mahdotonta havaita eroa alkuperäiseen osoitteeseen. Selainvalmistajat ovat kuitenkin reagoineet tammikuussa 2005 havaittuun haavoittuvuuteen, jotta käyttäjän olisi mahdollista huomata joutuneensa hyökkäyksen kohteeksi. [19]




Kuva 3.2: Tietovirran kulku välimieshyökkäyksessä.

### 3.3.3 Sovelluksen turvallisuus

Ohjelmiston toteutuksen aikana on kehittäjällä mahdollisuus vaikuttaa ainoastaan sovelluksen sisäiseen tietoturvaan. Niinpä tähän asiaan kiinnitettiin huomiota jo suunnitteluvaiheessa, jotta toteutettavan sovelluksen tietoturvataso on alusta asti riittävä.

Tietyllä tapaa tarkasteltuna lähdekoodin avoimuus voi heikentää järjestelmän tietoturvaa. Hyökkääjä on mahdollisuus tutustua sovelluksen rakenteeseen ja mikäli hän havaitsee ohjelmistossa mahdollisesti piileviä tietoturva-aukkoja, voi hän hyökätä niitä hyväksikäyttämällä. Niinpä tämä on pyrittävä estämään rakentamalla järjestelmästä riittävän turvallinen.

#### Istuntojen tietoturva

Istuntopohjaiset sovellukset, jossa käyttäjä joutuu autentikoimaan itsensä ennen varsinaisen käytön aloittamista, sisältävät haasteita sovelluskehittäjän näkökulmasta. Onnistuneen sisäänkirjautumisen jälkeen käyttäjälle luodaan istunto, jolloin kyetään kontrolloimaan hänen pääsyään rajattuihin toimintoihin ja tietoihin. Tämä istunto on kyettävä suojaamaan niin, ettei ulkopuolisella ole mahdollisuutta tarkastella tai muokata tietoja, joihin hänellä ei ole oikeutta.

Istuntopohjaisen WWW-sovelluksen tietoturvaa toteutettaessa on tietoturvaa mahdollista pyrkiä parantamaan seuraavien seikkojen avulla [20]:

- Sidotaan istuntoavain palvelimen saamaan selaimen verkko-osoitteeseen
- Sidotaan istuntoavain käyttäjän SSL sertifikaattiin
- Tuhotaan istunto uloskirjautuessa tai palvelimen toimesta tietyn ajan toimimattomuuden jälkeen
- Kehotetaan käyttäjiä istunnon päätteeksi kirjautumaan ulos
- Käytetään kiinteää istunnon kestoa, jonka jälkeen autentikointi vaaditaan uudelleen

Istuntojen tarkkailussa IP-osoitetta voidaan käyttää yhtenä parametrina. Osoite saattaa kuitenkin muuttua kesken istunnon esim. NAT:in (Network Address Translation) seurauksena ja osaava hyökkääjä pystyy joka tapauksessa kiertämään tämän suojauksen halutessaan, joten sen tuoma hyöty ei ole järin suuri. Myös istunnolle määritettävä kiinteä enimmäiskesto voi muodostua käyttäjän kannalta melko epämiellyttäväksi.

Istuntoavaimen vuotaminen ulkopuoliselle taholle voi aiheuttaa suuren tietoturvariskin. Saadessaan tietoonsa voimassa olevan istuntoavaimen, hyökkääjällä on mahdollisuus suorittaa kaikki istunnon varsinaisen omistajan käytössä olevat toiminnot hänen nimissään.

Istuntoja vastaan suoritettavat hyökkäykset voidaan jakaa kahteen varsinaiseen pääryhmään. Ensimmäinen niistä perustuu olemassaolevan istunnon kaappaamiseen, johon voidaan laskea kuuluviksi yhteyden sieppaaminen, avaimen ennustaminen ja avaimen arvaus. Toinen tapa on kiinteän tunnisteen hyökkäys. Taulukossa 3.2 on esitetty ratkaisut, joiden avulla pyritään hyökkäysten estämiseen.

Taulukko 3.2: Istuntoja vastaan suoritettavien hyökkäysten torjuminen.

| **Hyökkäystapa** | **Ongelman ratkaisu** |
|---|---|
| Yhteyden sieppaaminen | Tietoa kuljetetaan verkon yli ainoastaan salattuna, käyttäjän henkilökohtainen tietoturva (esim. virustorjunta). |
| Ennustaminen | Luodaan vahvasti sattumanvarainen istuntoavain. |
| Arvaus (Brute Force) | Luodaan riittävän pitkä istuntoavain. |
| Kiinteän tunnisteen hyökkäys | Luodaan jokaiselle istunnolle uusi istuntoavain. |

Yhteyden sieppaaminen tapahtuu urkkimalla tietoon voimassa oleva istuntoavain. Tämä voi tapahtua salakuuntelemalla yhteyttä ja kaappaamalla istuntoavain sieltä tai selvittämällä se käyttäjän keksistä. Avaimen ennustaminen on mahdollista, mikäli sen luonti ei ole riittävän sattumanvaraista. Yksi räikeimmistä esimerkeistä voisi olla käyttää istuntoavaimena lukuarvoa, jonka arvoa kasvatetaan yksi kerrallaan. Arvaaminen taas on sitä helpompaa, mitä lyhyempi luotava avain on.

Kiinteän tunnisteen hyökkäys perustuu siihen, että hyökkäävä osapuoli pyrkii vaikuttamaan istuntoavaimen muodostamiseen. Tämä on mahdollista niin, että käyttäjälle onnistutaan välittämään linkki kyseiseen palveluun, usein joko ulkopuolisen WWW-sivuston, sähköpostiviestin tai käyttäjän koneelle tallennetun keksin tietojen muokkaamisen avulla. Annetussa linkissä istuntoavain on valmiiksi määritettynä. Kun käyttäjä kirjautuu onnistuneesti järjestelmään, istunnolle ei luoda uutta avainta, vaan otetaan käyttöön käyttäjän lähettämä arvo. Tämän ansiosta hyökkääjän tarvitsee ainoastaan odottaa, että uhri kirjautuu tunnuksellaan sisään, jonka jälkeen hänellä on yhtäläinen mahdollisuus järjestelmän käyttöön istunnon voimassaolon ajan.

PHP tarjoaa sisäisiä funktioita istuntojen ylläpitämiseen, joiden avulla istuntojen hallintaa pyritään sovelluskehittäjän kannalta helpottamaan. Oletuksena PHP käyttää keksejä tämän istuntojen tiedon tallentamiseen. Keksien käyttö vaatii kuitenkin, että käyttäjän selain tukee niitä ja käyttäjä on hyväksynyt niiden käytön. Mikäli keksien käyttöä ei tueta, osaa PHP automaattisesti sisällyttää tiedot automaattisesti sekä POST- että GET-metodeihin perustuviin pyyntöihin. Istuntoavaimen sisällyttäminen GET-pyynnöissä URL-osoitekenttään voi kuitenkin aiheuttaa tietoturvariskin, mikäli se automaattisesti sisällytetään myös ulkopuolisiin sovelluksiin viittaaviin linkkeihin. Tällöin ulkopuoliseen resurssiin viittaava linkki noudattaa muotoa:

```
http://my.server.com/?sessionID=asu4G8u34Xg83nmgS
```

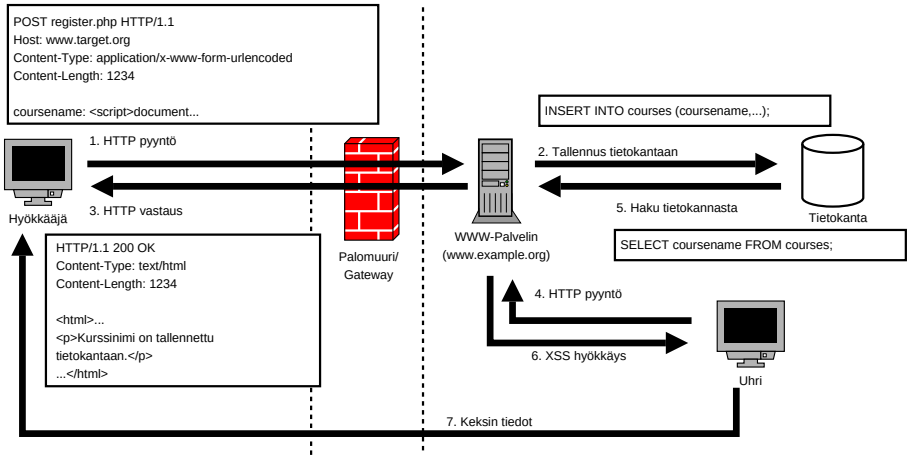
Mikäli käyttäjä suorittaa pyynnön linkitettyyn resurssiin, ulkopuolinen taho voi HTTP-pyynnöstä tai kyseisen palvelimen lokitiedoista lukea lähetetyn istuntoavaimen arvon. Tätä kautta avaimen haltuunsa saava taho voi sen avulla kaapata haltuunsa avoimen istunnon. Tämän takia PHP:n istuntojen toteutuksessa tulee varmistaa, että palvelusta ulospäin viittaaviin linkkeihin ei automaattisesti lisätä ylimääräistä tietoa, kuten istuntoavainta.

Mikäli hyökkääjällä on fyysisesti mahdollisuus nähdä osoitekenttä, voi hän käyttää samaa osoitekentässä näkyvillä olevaa istuntoavainta jonka avulla saa pääsyn palveluun. Tämä voidaan estää määrittämällä sessiohallinnan asetukset niin, että is-

tuntoavaimen välittämisessä ainoastaan keksien käyttö on sallittu. Tällöin palvelu kuitenkin toimii ainoastaan selaimilla, jotka sallivat keksien käytön.

Keksien sisältämät tiedot voidaan varastaa myös käyttäjän omalta päätteeltä. Mikäli hyökkääjällä on mahdollisuus päästä käsiksi keksin tiedot sisältävään tiedostoon esimerkiksi haittaohjelman avulla, voidaan istuntoavain lukea siitä suoraan. Tämän tietoturvauhkan estäminen jää käyttäjän omalle vastuulle, sillä palvelinpäässä tämän ongelman hallitseminen on mahdotonta. Käyttöjärjestelmän ja komponenttien sekä virustorjuntaohjelmiston ajan tasalla pitäminen parantaa käyttäjän itsensä tietoturvaa.

Keksin sisältämän tiedon urkkimiseen voidaan käyttää myös XSS-hyökkäystä (Cross-Site Scripting). Tällöin palveluun on saatu syötettyä tietoa, joihin on pääsy myös muilla käyttäjillä [21]. Sovellus on altis tälle riskille, mikäli ulkopuolelta lähetettävään tietoon luotetaan automaattisesti, sille ei tehdä tarkastuksia ja se päätyy myöhemmin muiden käyttäjien saataville [22] [23]. Kuvassa 3.3 on esitetty lähteeseen [22] perustuva skenaario XSS-hyökkäyksen etenemisestä. Esimerkissä hyökkääjä on ensin vaiheissa 1-3 onnistunut tallentamaan järjestelmän tietokantaan tarkastamatonta tietoa. Myöhemmin uhri hakee palvelimelta tietoa, jolloin hyökkääjän tallentama tieto välittyy uhrille vaiheissa 6-7. Uhrin vastaanottama tieto sisältää automaattisesti asiakaspäässä suoritettavan skriptin, jolloin keksin tiedot välitetään hyökkääjälle uhrin sitä havaitsematta.




Kuva 3.3: Skenaario XSS-hyökkäyksen etenemisestä.

Toteutettavassa sovelluksessa potentiaalinen haavoittuvuus voisi löytyä tietokannasta puuttuvien kurssinimien kohdalta, jolloin käyttäjille annetaan mahdollisuus syöttää itse väliaikaiseen käyttöön tarkoitettu kurssinimi. Mikäli syötettä ei tarkasteta,

aiheutetaan riski toisille käyttäjille sekä ylläpidolle. Allaolevassa esimerkissä esitetään JavaScript-koodi, joka järjestelmään piilotettuna lähettää käyttäjän keksin toiselle palvelimelle. Tätä kautta ulkopuolisella olisi mahdollisuus saada istuntoavain ja samalla päästä käsiksi palveluun toisen käyttäjän oikeuksilla.

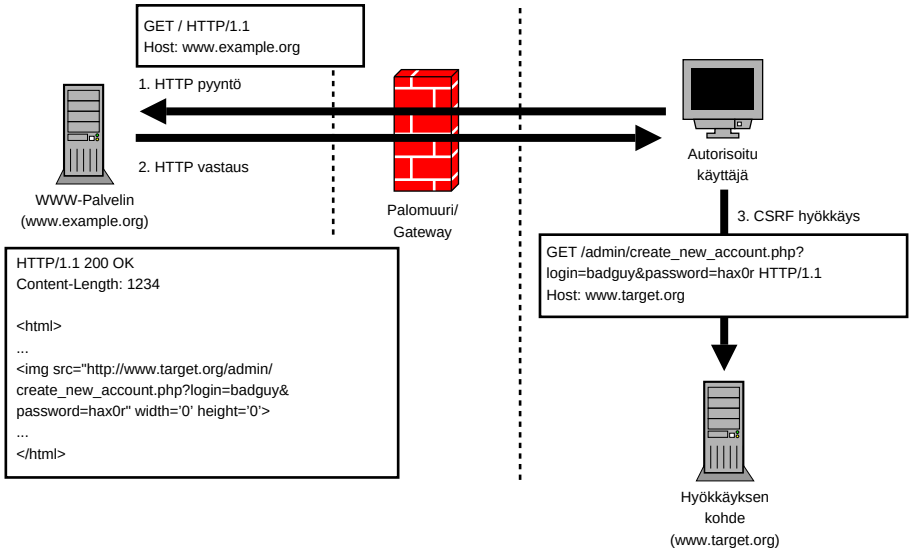
```
<script>
document.location = 'http://evil.server.org/steal_cookies.php?
cookies=' + document.cookie
</script>
```

Oman tietoturvauhkansa muodostaa myös CSRF-hyökkäys (Cross-Site Request Forgeries). XSS-hyökkäykseen verrattuna se on lähes päinvastainen hyökkäystapa. Hyökkäyksessä ei niinkään hyödynnetä käyttäjän luottamusta WWW-sovellusta kohtaan, vaan sovelluksen luottamusta käyttäjää kohtaan. [22] Uhrilla olevaa käyttöoikeutta rajattuun järjestelmään käytetään hyväksi ja saadaan hänet huomaamattaan suorittamaan haluttu toiminto tämän avulla. Allaolevassa esimerkissä hyökkääjä on sisällyttänyt kuvan dokumenttiin, joka on kooltaan olematon ja näin näkymättömissä käyttäjälle. Tämä dokumentti on saatu välitettyä kohteelle mahdollisesti joko sähköpostin tai toisen verkkopalvelun avulla. Hyökkäyksen toimivuuden ehtona on, että käyttäjällä on oikeus toiminnon suorittamiseen eli hän on kirjautuneena hyökkäyksen kohteena olevaan palveluun ja samaan aikaan suorittaa hyökkääjän lähettämän pyynnön resurssiin.

```
<img src='http://www.target.org/admin/create_new_account.php?
login=badguy&password=hax0r' width='0' height='0'>
```

Resurssin lähempi tarkastelu osoittaa, että kyse ei itse asiassa ole kuvatiedostosta, vaan se on viittaus ulkopuolisen palvelun toimintoon, jonka avulla hyökkääjä luo itselleen käyttäjätunnuksen kyseiseen palveluun. CSRF-hyökkäyksen toimintaperiaate ja edellämainitun haavoittuvuuden hyödyntäminen verkkotasolla on esitetty kuvassa 3.4. Käyttäjän hyödyntämään resursiin (www.example.org) on tallennettu linkki ulkopuoliseen resurssiin. Kun käyttäjä on ensin hakenut alkuperäiseltä palvelimelta dokumentin (vaiheet 1-2), suorittaa selain automaattisesti pyynnön tässä dokumentissa viitattuun resurssiin vaiheessa 3. Mikäli hyökkäyksessä hyödynnettävällä henkilöllä on kyseisellä hetkellä käyttöoikeus tähän resurssiin, suoritetaan haluttu toiminto.

Tämän toiminnon hyödyntäminen vaatii hyökkääjältä sovelluksen rakenteen ja toteutuksen tuntemisen. Tämäntyyppisen haavoittuvuuden riskejä voidaan hyväksymällä ainoastaan POST-metodilla lähetetyt lomakkeet. Kokonaan ongelmaa ei pelkästään




Kuva 3.4: CSRF hyökkäys. [22]

tällä kuitenkaan pystytä poistamaan, joten lisäksi voidaan tehdä muita tarkastuksia, kuten että lomake on alun perin ladattu oikeasta järjestelmästä.

### Tietokantatoimintojen tietoturva

Dynaamiset WWW-sovellukset, joiden toiminta perustuu tietokantojen käyttöön, sisältävät sovellustasolla tietoturvariskejä. Tietokannan muokkaamisen tulee olla mahdollista vain rajatulla ryhmällä, joiden toimia myös kyetään mahdollisuuksien mukaan kontrolloimaan. Jokainen huonosti kontrolloitu haku tietokantaan sisältää riskin, joka on kyettävä minimoimaan. Riski voidaan hallita kontrolloimalla käyttäjän syötteitä sekä rajaamalla sovelluksen oikeuksia suorittaa muutoksia tietokantaan.

Tietokantahaavoittuvuus, SQL-injection, aiheutuu, kun käyttäjien palvelimelle lähettämistä epäluotettavista arvoista muodostetaan tietokantakyselyitä ilman sisällön tarkempaa kontrollia. Ne saattavat aiheuttavat mielivaltaisten SQL-komentojen suorittamisen hyökkääjän toimesta. Tämä tekniikka, joka sallii tietokannan rajattoman manipuloinnin, aiheuttaa valtaosan WWW-pohjaisia sovelluksia vastaan onnistuneesti tehdyistä hyökkäyksistä. [23]

Yksinkertainen autentikointi voidaan suorittaa käyttämällä seuraavaa SQL-kyselyä:

```
SELECT username FROM user_table WHERE username='syöte1' AND
password='syöte2';
```

Esitetyssä kyselyssä *syöte1* on käyttäjätunnus ja *syöte2* on käyttäjän syöttämä salasana. Kuvitellaan, että hyökkääjä antaa salasanakseen syötteen

```
syöte2 = "1' OR '1'='1"
```

Tällöin suoritettava tietokantakysely olisi

```
SELECT username FROM user_table WHERE username='syöte1' AND
password='' OR '1'='1';
```

Mikäli autentikointimekanismi olisi toteutettu kuvatulla tavalla, jossa käyttäjän syötettä ei erikseen tarkasteta, palauttaisi kysely annetun käyttäjätunnuksen huolimatta siitä, onko annettu salasana oikea. Näin hyökkääjällä olisi mahdollisuus kirjautua järjestelmään tuntemalla ainoastaan olemassaoleva käyttäjätunnus. Esimerkki havainnollistaa, millainen tietokantaa vastaan suoritettavan hyökkäyksen toimintaperiaate on.

Huomattavasti edellistäkin esimerkkiä suurempaa vahinkoa voi saada aikaiseksi antamalla lomakkeeseen esimerkiksi syötteen

```
';DROP TABLE 'students
```

jolloin tietokantaan suoritetut komennot olisivat

```
SELECT username, password FROM user_table WHERE username='syöte1' AND
password='';

DROP TABLE 'students';
```

Ensimmäisenä esitetyn kyselyn suorituksen jälkeen suoritettaisiin käyttäjän lisäämä kysely. Tietokannan rakenteen tuntevalla hyökkääjällä olisi mahdollisuus tämän kyselyn avulla poistaa koko `students`-taulun sisältö. Edellä kuvatut esimerkit osoittavat, kuinka välttämätöntä käyttäjän lähettämän syötteen kontrolloiminen on.

Chris Shiflett on artikkelissaan [22] määritellyt viisi asiaa, jotka kannattaa ottaa huomioon käsiteltäessä käyttäjien lähettämää informaatiota:

- *Suodata kaikki ulkopuolinen tieto.* Kaikki ulkopuolinen tieto tulee tarkastaa ennen sen käsittelyä
- *Käytä olemassaolevia funktioita.* Käytä mieluummin valmiita funktioita, kuin että yrität itse luoda saman tehtävän suorittavan toiminnon.
- *Hyväksy ainoastaan soveltuva sisältö.* Aseta mieluummin liian tarkat kuin löy-

hät rajoitukset käsiteltävälle tiedolle.

- *Käytä tarkkaa nimeämistapaa.* Erottele sovelluksessa sisäisesti alkuperäiset ja suodatetut muuttujat.
- *Ole luova.* Älä tee oletuksia käyttäjän toiminnasta äläkä luota mihinkään.

Täysin turvallisen sovelluksen, jossa on palvelimen ja käyttäjän välistä interaktiivisuutta, toteuttaminen on haastava tehtävä. Aika ajoin havaittavat haavoittuvuudet vaativat myös sovellusten ylläpitoa ja päivittämistä. Tietoturvaan liittyvien seikkojen huomioiminen riittävän aikaisessa vaiheessa on kuitenkin edellytys verkkopohjaista sovellusta toteutettaessa.

## 3.4 Muut toteutukseen liittyvät järjestelmät

Ennen järjestelmän suunnittelua kartoitettiin lisäksi LTY:llä käytössä olevat tietojärjestelmät ja -palvelut, joita voidaan hyödyntää järjestelmää toteutettaessa. Järjestelmät voidaan jakaa käyttötarpeeltaan kahteen eri kategoriaan: toiset sisältävät järjestelmän toiminnan kannalta tarpeellista tietoa ja toisten tarjoamia palveluita pystytään hyödyntämään sovellusta käytettäessä.

Järjestelmän toiminnan kannalta tarpeellista tietoa ovat opintojaksojen tiedot ja tutkintorakenne. Sovellusta käyttävien opiskelijoiden kannalta myös opintosuoritustieto on tarpeellista. Kaikki tämä tieto on jossain muodossa tallennettuna jo olemassaolevissa järjestelmissä.

Näiden tietojen lisäksi käyttäjien tunnistaminen käyttäen jo olemassaolevia käyttäjätunnuksia nähtiin tarpeelliseksi. Tämä on sekä ylläpitäjän että opiskelijoiden kannalta paras mahdollinen tapa toimia.

### 3.4.1 Opintosuoritusten siirtäminen järjestelmään

LTY:ssa kirjoilla olevien opiskelijoiden suoritustiedot ovat tallennettu keskitetysti yliopistolla käytössä olevan Oodi-järjestelmän tietokantaan. Jotta järjestelmää käyttävät opiskelijat säästyisivät turhalta manuaaliselta työltä suorittamiaan kursseja järjestelmään kirjatessa, oli tavoitteena opintorekisteriotteen automaattinen kopioiminen järjestelmän tietokantaan. Opiskelijoiden jo suorittamien opintojen siirtäminen järjestelmään vaatii rajapintaa kyseisen tiedon sisältämän järjestelmän kanssa.

Alkuperäinen toive oli saada WebTUTORin ja Oodin välille Web Services -ratkaisuun pohjautuva rajapinta. Jo alun perin osasimme odottaa, että tämän ratkaisun hyödyntäminen diplomityöni toteutuksen aikana on varsin epätodennäköistä. Muutosten hyväksyttäminen ja toteuttaminen jo olemassaolevaan järjestelmään olisi vähintäänkin vaatinut huomattavan määrän byrokratiaa, silti vailla taetta onnistumisesta.

Toinen vaihtoehto opintosuoritusten siirtämiseksi automaattisesti järjestelmään Oodista oli suora tietokantayhteys. Aiemmin mainittuun Desmond-järjestelmään on tällainen toteutettu, jolloin sovellukselle on annettu oikeus lukea suoraan tietokannasta tiettyjen taulujen sisältöjä. Muutosoikeutta tietokannassa oleviin tietoihin ei tämän rajapinnan kautta ole. [24]

Helsingin yliopisto on maksanut korvauksen Oodiin toteutetun lisäosan kehittämisestä. Koska aiheutuneen kustannuksen katsottiin olevan niin pieni, oltiin tehty työ valmiita jakamaan LTY:n kanssa ilman erillistä korvausta. [8] LTY:n tietohallinnon mielipide kuitenkin oli, ettei ole tarvetta tukea järjestelmää, jonka käytöstä yliopistotasolla ei ole virallista päätöstä. [25] Niinpä suoritustietoa ei Oodista WebTUTORiin automaattisesti saada, vaan opiskelijat joutuvat lähettämään opintosuorituksensa järjestelmään erillisenä, Oodista tallennettuna tai sähköpostitse tilattuna tiedostona.

### 3.4.2 Opintojaksojen ja tutkintorakenteiden tallentaminen järjestelmään

Järjestelmään oli sen toiminnan kannalta tallennettava tarjolla olevien opintojaksojen tiedot. Koska tietojen tuominen Oodista ylipäänsä oli osoittautunut käytännössä mahdottomaksi, oli kurssitieto kopioitava jostain muualta mahdollisimman pienellä vaivalla. Tietoa opintojaksoista oli kuitenkin onneksi saatavilla useassa eri paikassa, joten päätimme kopioida tiedot LTY:n omasta Sotka-tietojärjestelmästä.

Sotka on tietojärjestelmä opinto-oppaan tuottamisprosessin tueksi, joka koostuu opinto-opastietokannasta ja sen julkaisujärjestelmästä. [26] Se on Microsoft Access-pohjainen sovellus, joka alun perin sisälsi SQL-tietokannan, käyttöliittymän lomakkeet sekä ohjelmakoodin. Myöhemmin sovellusta on kehitetty niin, että tietokanta on keskitetty Microsoft SQL Server -tietokantapalvelimelle. Järjestelmän avulla saadaan vähennettyä huomattavasti manuaalisen työn määrää opinto-oppaan lopullisessa koostamisvaiheessa ja samalla kaikkien osastojen tuottamille osille saadaan automaattisesti tuotettua yhtenäinen ulkoasu.

Sotka on toteutettu niin, että sen tietokanta sisältää ainoastaan viimeisimmän opinto-

oppaan tiedot. Niinpä siitä puuttuu suurin osa aiemmin opetetuista kursseista, joita ei enää järjestetä. Lisäksi tietokanta on optimoitu opinto-oppaan julkaisemista silmälläpitäen, ei niinkään kurssien tai tutkintorakenteen näkökulmasta. Niinpä ainoastaan kurssitietoa on mahdollista järkevästi hyödyntää järjestelmässämme.

Työn toteuttamisen kannalta oli Sotkan sisältämä kurssitieto kuitenkin luontevaa kopioida WebTUTOR-järjestelmään. Järjestelmään tuotiin opintojakson tunnus ja sekä suomen- että englanninkielinen nimi ja opintoviikkomäärä. Tiedot kopioitiin kertaluonteisesti ODBC-rajapinnan avulla Sotkan tietokannasta WebTUTORin kantaan. Tarvittaessa järjestelmän kurssikanta voidaan päivittää myöhemmin.

Sotka tulee tuskin olemaan kovin pitkäikäinen ratkaisu yliopiston käytössä ja siihen kohdistuu melko vahvoja uudistuspaineita. [27] Tulevaisuudessa kurssitieto on todennäköisesti siirrettävä korvaavasta järjestelmästä. Yksi mahdollisuus voisi olla meneillään oleva eOPAS-hanke[1].

Tutkintorakenteesta ei toistaiseksi ole saatavilla riittävän rakenteista tietoa, jota olisi voitu hyödyntää toteutuksen puitteissa. Niinpä kaikki järjestelmään tuotavat rakenteet on luotava manuaalisesti. Tätä varten on järjestelmään toteutettava käyttöliittymä, jonka avulla se voidaan tehdä. Tarkempi kuvaus tutkintorakenteen muodostamisesta on esitetty kappaleessa 4.2.1.

### 3.4.3 Käyttäjätunnistus RADIUS-palvelun avulla

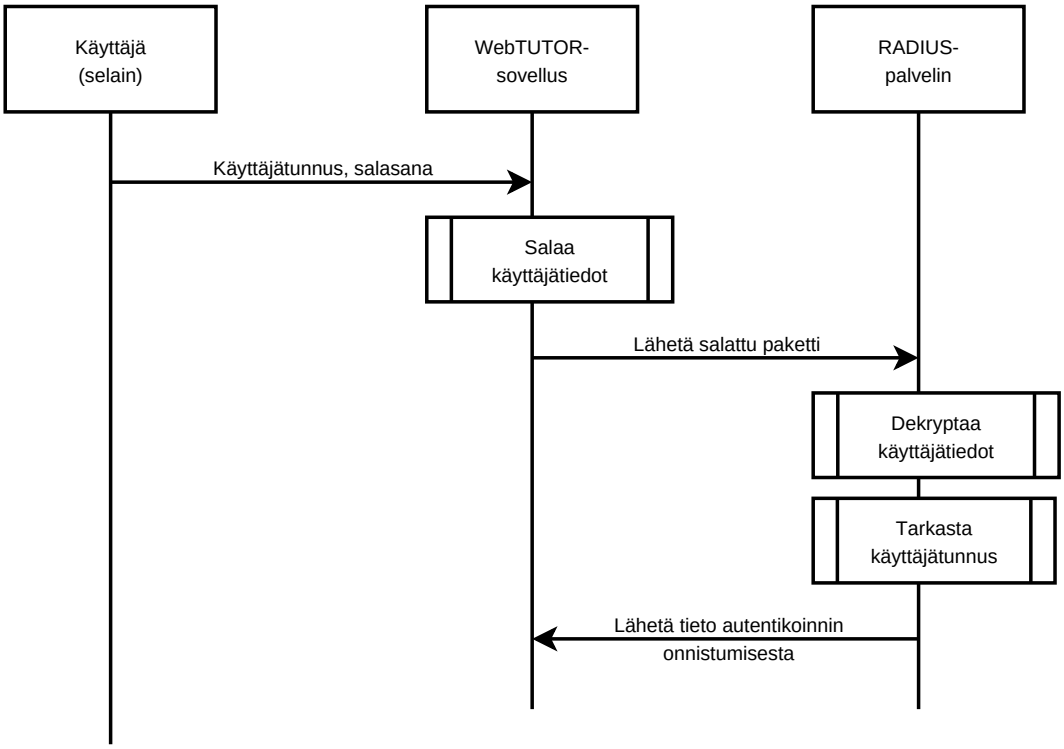
LTY:llä on käytössään RADIUS-palvelin, jota hyödynnetään käyttäjien autentikoinnissa. Sen ylläpidosta vastaa hallintopalvelukeskuksen henkilökunta. Sovelluksellemme annettiin oikeus käyttää yliopiston RADIUS-palvelinta käyttäjien autentikointiin, jolloin toteutettava sovellus on kaikkien yliopiston käyttäjätunnuksen omaavien käytössä ilman erillistä tunnuksen luomista.

RADIUS-tunnistus pohjautuu palvelin/asiakas-pohjaiseen arkkitehtuuriin. Käyttäjän tunniste lähetetään palvelimelle salattuna, jonka tiivistealgoritmin syötteen osana on käytetty asiakkaan ja palvelimen välille sovittua salaista avainta. RADIUS on määritelty RFC 2138:ssa. [28] Tarkemmin rajapinnasta on kerrottu kappaleessa 5.7.1.

Kuvassa 3.5 on esitetty RADIUS-palvelun avulla suoritettu autentikointi. Käyttäjä lähettää tunnuksen ja salasanan sovelluspalvelimelle, joka toimii tunnistusprosessissa

[1] `http://www.virtuaaliyliopisto.fi/?node=vp_kokousymparisto_fin&action=focus&kokousryhma=7`

asiakkaana. Käyttäjän syöttämät tiedot salataan käyttäen sovelluksen ja RADIUS-palvelimen yhteistä salaista avainta. Salattu informaatio lähetetään Radiukselle, joka purkaa tiedon, tarkastaa tunnusten oikeellisuuden ja palauttaa sovelluspalvelimelle tiedon autentikoinnin onnistumisesta. Tämän jälkeen sovellus suorittaa paluuarvoon sidotut toiminnot, kuten luo tarvittaessa käyttäjälle uuden istunnon.




Kuva 3.5: RADIUS-tunnistuksen vaiheet.

# Luku 4

# Järjestelmän suunnittelu

Projektin määrittelyvaiheessa asetettujen vaatimusten pohjalta järjestelmä oli tarkoitus toteuttaa soveltuvaksi LTY:n tietotekniikan osaston käyttöön. Myöhemmin kuitenkin päätettiin suunnitella ja toteuttaa koko järjestelmä niin, että se on tarvittaessa mahdollista ottaa koko yliopiston käyttöön. Samalla suunnittelussa päätettiin ottaa huomioon järjestelmän myöhempi laajennettavuus ja tuleva kehitystyö sekä mahdolliset rajapinnat muihin järjestelmiin. Tutkintorakenteen muuttuminen kaksivaiheiseksi ja samalla opintoviikkojen muuttuminen ECTS:n mukaisiksi opintopisteiksi tuli myös huomioida jo suunnitteluvaiheessa.

Järjestelmän suunnittelu voitiin jakaa toisistaan erillisiin osiin. Ensimmäisessä vaiheessa aiemmin toteutetun kartoitustyön perusteella pystyttiin kartoittamaan mahdollisuudet erilaisille rajapinnoille ja tätä kautta hahmottamaan järjestelmän arkkitehtuuri kokonaisuudessaan. Koska kaikki sovelluksen toimintaan käytettävä informaatio tallennetaan tietokantaan, toisessa vaiheessa oli tehtävänä riittävän laajan ja määrittelyn asettamat vaatimukset täyttävän tietokannan suunnittelu. Kolmas erillinen vaihe oli käyttöliittymälle määritetyn käytettävyyden suunnittelu toteutettavaksi käytännössä.

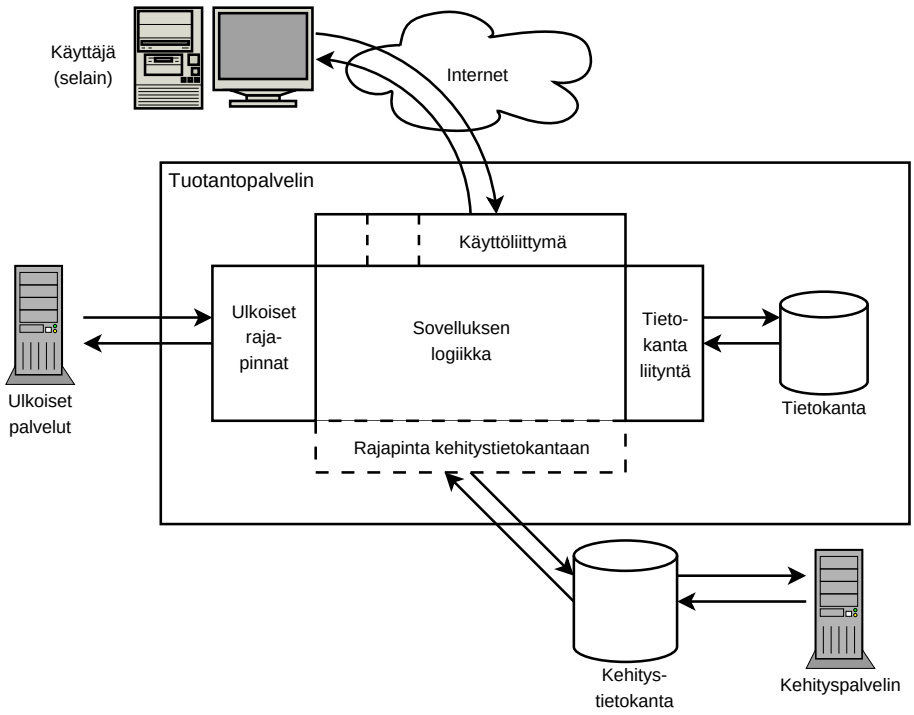
## 4.1 Arkkitehtuuri

Järjestelmän arkkitehtuuria suunniteltaessa sovelluksen toiminnot ja sille asetetut toiminnalliset vaatimukset jaettiin erillisiin osiin. Sen sisältämät tiedot tallennetaan tietokantaan, jonka sisältämän tiedon ympärille järjestelmän toiminnallisuus rakennetaan. Tietokantarajapinnan kautta tiedot siirretään sovelluksen käyttöön.

Järjestelmän toiminnallisesta puolesta vastaa sovelluksen logiikka. Se käsittelee tiedon ja suorittaa tarvittavat operaatiot tietoa käsiteltäessä. Järjestelmää käytetään WWW-käyttöliittymän avulla, joka havainnollistaa järjestelmän sisältämän informaation käyttäjälle.

Järjestelmä sisältää osia, jotka toimivat rajapintoina ulkopuolisiin järjestelmiin. Esimerkki tällaisesta rajapinnasta on RADIUS-tunnistus. Lisäksi järjestelmä sisältää rajapinnan mahdolliseen kehitystietokantaan. Yleisestiottaen on järkevää erotella tuotanto- ja kehitystietokannat ja -palvelimet toisistaan, jotta järjestelmän tietokantaan tehdyt muutokset pystytään testaamaan loppukäyttäjiltä eristetyssä turvallisessa ympäristössä. Kun myöhemmin vaadittava testaus on kehityspalvelimella saatu suoritettua, voidaan uusi tieto siirtää automaattisesti tuotantokäyttöön.

Koko järjestelmän toiminta perustuu kuvassa 4.1 on esitettyyn arkkitehtuuriin. On kuitenkin huomattava, että eri oikeudet omaaville käyttäjille on toteutettu omat, erilliset, käyttöliittymänsä palveluun. Niinpä esimerkiksi opiskelijoiden käyttöliittymästä ei ole pääsyä kehitystietokannan rajapintaan.




Kuva 4.1: Järjestelmän arkkitehtuuri ja komponenttien välinen tietovirta.

## 4.2 Tietokanta

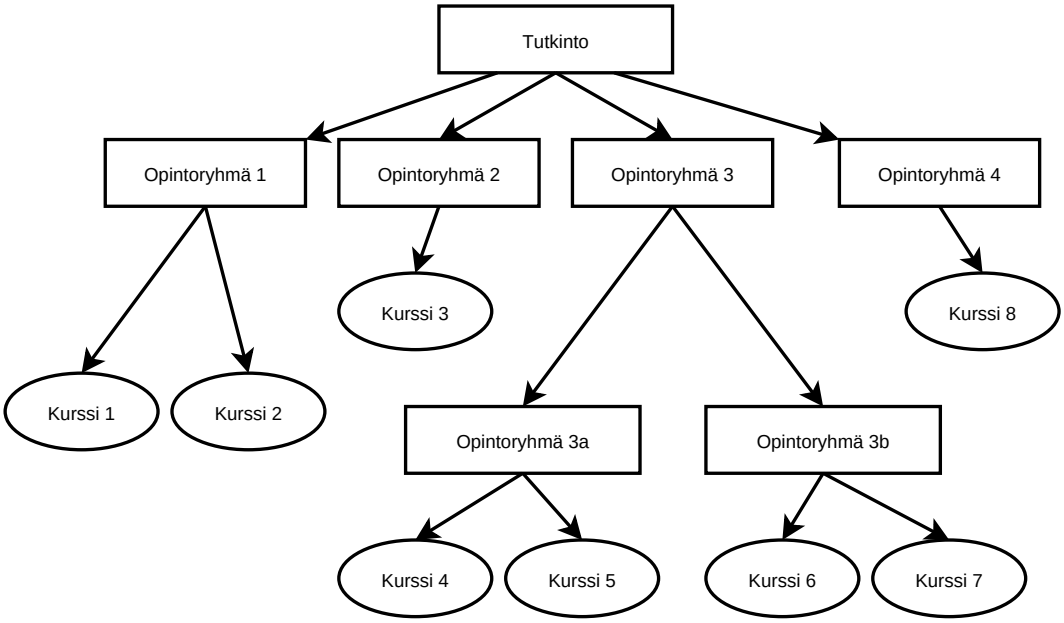
Järjestelmää suunniteltaessa pyrittiin sovelluksen rakenne toteuttamaan niin, että kaikki tutkintorakennetieto sekä siihen liittyvät poikkeukset pystytään kuvaamaan relaatiotietokannassa. Tämän avulla haluttiin saavuttaa tilanne, että myöhemmin tutkintojen rakenteissa mahdollisesti tapahtuvien muutosten yhteydessä lähdekoodiin ei tarvitse tehdä muutoksia tai ne ovat minimaalisia. Niinpä tietokannasta oli tehtävä riittävän laaja, jotta sen avulla kyetään kuvaamaan myös mahdolliset poikkeustapaukset. Näin myös pelkästään tietokannan rakenteen riittävällä tuntemuksella kyetään järjestelmästä saamaan ulos kaikki tarvittava informaatio.

Tietokannan taulut voidaan pääpiirteittäin jakaa kolmeen erilliseen ryhmään: käyttäjätietotauluihin, tutkintorakennetauluihin ja opiskelijan HOPSiin liittyviin tauluihin. Käyttäjätietotauluihin kuuluu sovelluksen käyttäjien ja heidän yksittäisten istuntojen hallintaan tarvittavat taulut sekä käyttäjien väliset sidokset, kuten tarvittaessa tieto kullekin opiskelijalle erikseen määrätystä opintosuunnitelman tarkastajasta. Tutkintorakennetaulut sisältävät organisaation tutkintorakennetiedot yksittäisestä opintojaksosta aina tutkintoon asti. Opiskelijoiden HOPSin määrittäviin tauluihin tallennetaan opiskelijan valitseman tutkinnon rakenne sekä siihen liitetyt opintojaksot. Tietokantojen relaatiot on tarkemmin esitetty liitteessä 2.

### 4.2.1 Tutkintorakenne- ja kurssitieto

Tutkintorakennetta mallinnettaessa päädyttiin käyttämään ns. puumallia. Tämä rakenne antaa riittävän hyvät mahdollisuudet sekä aiemman että nykyisen, kaksiportaisen, tutkintorakenteen kuvaamiseen riittävän tarkasti ja hallittavasti. Yksinkertaistettu esimerkki hierarkkisesti kuvatusta tutkintorakenteesta on esitetty kuvassa 4.2. Rakenteen muodostaminen aloitetaan luomalla tutkinto, johon liitetään tarvittava määrä erillisiä opintoryhmiä. Näillä opintoryhmillä voi olla omia aliryhmiään ja niihin voidaan liittää opintojaksoja. Sekä opintokokonaisuudet että opintojaksot voidaan määrittää joko pakollisiksi tai vapaaehtoisiksi. Se, että tutkintorakenne sisältää opintoryhmien lisäksi myös opintojaksot, auttaa myöhemmin opiskelijaa huomattavasti suunnitelmansa laatimisessa.

Jokainen opintoryhmä on sidottu joko yksittäiseen osastoon tai opintosuuntaan. Perusopinnot ovat yleensä yhteisiä kaikille osaston opiskelijoille, mutta syventävät opinnot riippuvat valitusta opintosuunnasta. Tämäntyyppisellä ryhmien keskinäisellä jaottelulla mahdollistetaan se, että opiskelijan tutkintorakenne automaattisesti




Kuva 4.2: Opintorakenteen hierarkkinen mallinnus.

pohjautuu oman opintosuunnan rakenteeseen.

Koska opiskelija voi valita tutkintoonsa sisällytettävät opintoryhmät eri vuosien tutkintorakenteista, ovat myös eri vuosilta olevat keskenään vaihtoehtoiset ryhmät sidottava toisiinsa. Tämä suunniteltiin toteutettavaksi niin, että jokaiselle tutkintoon sisällytetylle opintokokonaisuudelle on määrätty oma tunniste. Eri vuosilta olevat keskenään vaihtoehtoiset opintokokonaisuudet liitetään toisiinsa tämän tunnisteen perusteella.

Järjestelmän sisältämät kurssit yksilöidään kurssitunnisteen perusteella. Kurssitiedot sisältävät lisäksi suomen- ja englanninkielisen nimen sekä opintoviikko- ja ECTS opintopistemäärät. Koska opintojaksoille ei vielä niitä järjestelmään siirrettäessä ollut päätetty opintopistemäärää, ECTS pisteiden määrittämiseen käytettiin kerrointa 1,5 opintoviikkomääriin verrattuna. Kun myöhemmin opintopistemäärät kursseille määritetään, voidaan ne tarvittaessa päivittää järjestelmään. Samalla tapahtuva kurssitunnisteiden muutos vaatii joka tapauksessa, että kurssitiedot on kokonaisuudessaan tuotava uudelleen järjestelmään.

Järjestelmän tutkintorakennetietokantaan voidaan tallentaa tietoa opintojaksojen välisistä ryhmäkohtaisista vaihtoehtoisuuksista, jolloin vain yhden ryhmään kuuluvan opintojakson suoritus vaaditaan. Esitietovaatimuksia tai -suosituksia voidaan hyödyntää opintosuunnitelmia arvioitaessa. Tiedon mallintamisella voidaan tarkas-

taa, että valittujen kurssien esitietovaatimukset täyttyvät ja että esitiedot ovat joko suoritettu tai aikataulutettu suoritettavaksi ennen niitä vaativan kurssin suorittamista.

### 4.2.2 Opiskelijan HOPS

Opiskelijan itselleen luoman suunnitelman sisältämä tietokanta vastaa rakenteeltaan hyvin paljon alkuperäisen tutkintorakenteen mallinnukseen käytettyä rakennetta. Sitä on kuitenkin voitu tietyiltä osin yksinkertaistaa. Suuri osa tallennettavasta rakenteesta sisältää viitteitä alkuperäisen tietokannan tietoihin.

Vuosien saatossa samansisältöisten ja toisiaan vastaavien kurssien tunnisteet voivat muuttua. Tällaisissa tapauksissa opiskelija voi merkitä opintosuunnitelmaansa sisällyttämän kurssin vastaamaan tutkintorakenteessa suoritettavaksi vaadittavaa kurssia. Korvaavuutta vastaavasti käytetään, kun opintosuunnitelmassa oleva opintojakso haetaan korvattavaksi toisella suorituksella.

### 4.2.3 Sovelluksen käyttäjätiedot

Sovelluksen sisältämään käyttäjätietoon sisältyy sekä henkilökunnan että mahdollisesti paikalliseen tietokantaan tallennettujen käyttäjätunnusten tiedot. Myös sovelluksen istuntojen hallintaan käytettävät taulut voidaan laskea kuuluvaksi tähän ryhmään.

Yksittäiselle opiskelijalle voidaan määrätä henkilökohtainen opettajatutor, jolla on oikeus lukea ja tarkastaa opiskelijan laatima HOPS. Myös tämän tiedon kuvaaminen kuuluu osaksi käyttäjätietojen mallinnusta. Näihin tietoihin tulee olla pääsy ainoastaan järjestelmän ylläpitäjällä.

## 4.3 Käyttäjien roolit

Järjestelmän käyttäjät voidaan jakaa rooleiltaan kolmeen ryhmään. Opiskelijat suunnittelevat työkalulla opintonsa, opintojen ohjaajat käyttävät sitä ohjaamisen työvälineenä suunnitelmia tarkastaessaan ja järjestelmän ylläpitäjien rooli on huolehtia järjestelmän toiminnasta, tutkintorakenteiden sisällöstä ja tiedon eheydestä. Jokaisen ryhmän vaatimukset ja niille tarjolla olevat toiminnot erotellaan toisistaan erillisiksi ryhmiksi, joille jokaiselle toteutetaan oma käyttöliittymänsä.

Kun käyttäjien roolit ja tehtävät on riittävän selkeästi määritelty, voidaan tämän tiedon avulla sovellusta suunniteltaessa ja toteutettaessa tehdä perusteltuja päätöksiä. Tällaisiin päätöksiin voidaan pääasiassa laskea kuuluviksi erilaiset käytettävyyteen liittyvät ratkaisut.

### 4.3.1 Opiskelijat

Järjestelmän varsinainen kohderyhmä on opiskelijat. Jokainen opiskelija joutuu jossain opintojensa vaiheessa laatimaan suunnitelman opinnoistaan, joka noudattaa sille tutkintorakenteessa asetettuja vaatimuksia. Käytännössä taulukkolaskenta- tai tekstinkäsittelyohjelmalla toteutetut opintosuunnitelmat ovat varsin kertakäyttöisiä ja usein niiden hankalana koetun päivittämisen sijaan on myöhemmin laadittu kokonaan uusi suunnitelma.

Opintosuunnitelman laatiminen voidaan jakaa erillisiin osiin. Ensin tehdään suuremmat valinnat, kuten opintosuunta, jonka jälkeen suunnitelmaa tarkennetaan aina opintojaksotasolle asti. Järjestelmää käyttävän opiskelijan tehtävät voidaan jakaa kuuteen pääryhmään:

- *Henkilökohtaisten tietojen hallinta.* Järjestelmän tulee sisältää perustiedot jokaisesta käyttäjästä. Näiden tietojen perusteella opiskelijan laatimalle opintosuunnitelmalle voidaan asettaa rajat ja samalla ohjastaa hänet mahdollisimman hyvin itselleen sopivan suunnitelman tekoon. Perustietoihin kuuluvat käyttäjän henkilökohtaiset tiedot sekä tieto opintosuunnasta, opintojen alkamisesta ja käytettävästä tutkintorakennepohjasta.

- *Tutkintorakenteen luominen.* Itse opintosuunnitelman laatiminen aloitetaan valitsemalla tutkintorakennepohja, jonka perusteella suunnitelma luodaan. Osa ryhmistä, kuten sivuaineopintoryhmät, ovat vapaasti valittavissa, jolloin opiskelijan päätettäväksi jää, mitkä kokonaisuudet hän haluaa tutkintoonsa sisällyttää. Koska eri osastot noudattavat erilaista käytäntöä opintokokonaisuuksien vaihtamisen ja valinnan suhteen, ei muokkaamista pyritä suuremmin rajoittamaan, vaan osastokohtaisten sääntöjen noudattaminen osin pakostakin jätetään opiskelijan vastuulle. Valittua tutkintorakennepohjaa voidaan halutessa muokata myöhemmin.

- *Opintojen syöttäminen ja niiden sijoittelu osaksi tutkintoa.* Kun tutkintorakenteeseen liittyvät valinnat on tehty, on vuorossa suoritettujen opintojen liittäminen osaksi opintosuunnitelmaa. Opiskelija tuo jo suorittamansa opinnot

järjestelmään ja sijoittaa ne oikeisiin opintoryhmiin. Tässä vaiheessa opiskelijan mekaanista työtä voidaan pyrkiä vähentämään sijoittamalla syötetyt opintojaksot automaattisesti valittuun rakenteeseen siihen soveltuvin osin. Loput opintojaksot käyttäjä itse sijoittaa haluamiinsa ryhmiin mahdollisten vastaavuuksien tai korvaavuuksien avulla.

- *Opintojaksojen valinta ja suunnittelu.* Kun suoritetut opinnot on sijoitettu tutkintorakenteeseen, lisätään opintosuunnitelmaan kurssit, jotka ovat vielä suorittamatta. Koska järjestelmän tietokannassa oleva tutkintorakenne sisältää myös opintoryhmien sisältämät kurssit, tarjotaan käyttäjälle yksinkertainen luettelo, josta hänellä on mahdollisuus valita haluamansa kurssit. Tutkintoon kuuluvat pakolliset, vapaavalintaiset ja vaihtoehtoiset kurssit havainnollistetaan toisistaan eriäviksi, jolloin opiskelijalla on suhteellisen helppo työ valita suunnittelemansa opintojaksot. Kun suunnitellut opintojaksot on valittu osaksi HOPSia, voidaan niiden suorittaminen aikatauluttaa.

- *Yhteenveto opintosuunnitelmasta.*: Luotu opintosuunnitelma voidaan tulostaa. Suunnitelluista opinnoista on myös mahdollista saada yhteenveto, josta näkee lukukausittain suunnitellun kuormittavuuden.

- *Suunnitelman julkaisu.* Opiskelijalla on mahdollisuus julkaista valmiiksi laadittu opintosuunnitelma ja lähettää se opintosihteerin arvioitavaksi sähköisesti. Luonnollisesti opiskelijalla on myöhemmin mahdollisuus muokata suunnitelmaansa ja päivittää sitä aina tarvittaessa.

Työkalun painotusalue on erilainen, kun käyttäjänä on joko opintojensa alussa tai valmistumisensa kynnyksellä oleva opiskelija. Aloittelevalla opiskelijalla on pääasiassa tarve valita suoritettavaksi aiottuja ja tutkintorakenteeseen sisällytettäviä kursseja sekä aikatauluttaa niitä. Paljon suorituksia omaavalla opiskelijalla pääpaino on selkeästi opintojen sijoittamisella tutkintorakenteeseen, jolloin sovellus on enemmänkin tutkintorakenne- kuin suunnittelutyökalu.

Opintosuoritusten kertyessä suunnitelma ja sen tila elää koko ajan. Askel askeleelta alkuperäisestä opintosuunnitelmasta muotoutuu ja kasvaa valmistumista ennen hyväksytettävä tutkintorakenne. Samalla näiden dokumenttien välinen ero hämärtyy.

### 4.3.2 Opintosihteerit

Opiskelijoiden lisäksi toinen järjestelmän käytöstä selkeästi hyötyvä ryhmä on opintosihteerit. Heille järjestelmä tarjoaa pääsyn vastuulleen määrättyjen opiskelijoiden

julkaistuihin opintosuunnitelmiin. Järjestelmää voidaan käyttää opintosihteerin kannalta ohjausta tukevana toimintona, jonka avulla kyetään tekemään nopeita alustavia arvioita suunnitelman mielekkyydestä.

Opintosihteerin tehtäviin järjestelmässä pääasiallisesti kuuluu:

- *Tulostaa raportteja opiskelijan HOPSista.* Opintosuunnitelmaan on heille tarjolla erilaisia näkymiä eri toimintoja varten. Tulostamista ja nopeaa läpikäyntiä silmälläpitäen on opiskelijan omaa näkymää vastaava liittymä. Lisäksi suunnitelmasta voidaan luoda erilaisia yhteenvetoja arvioinnin tueksi.
- *Tarkastaa opiskelijoiden HOPSeja.* Opintosihteerillä on mahdollisuus tarkastella vastuualueelleen määrättyjen opiskelijoiden julkaisemia HOPSeja. Opintosuunnitelmaa voi tarkastella lähemmin ja saada siitä käyttäjältä piilotettua tarkempaa tietoa. Tällainen tieto koskee esimerkiksi tapaa jolla opintosuoritus on alun perin siirretty järjestelmään ja onko suoritus järjestelmän automaattisesti vai opiskelijan itsensä manuaalisesti tutkintorakenteeseen sijoittama. Automaattisesti järjestelmän tutkintorakenteeseen sijoittelemat kurssit ovat mitä suurimmalla todennäköisyydellä sijoitettu sääntöjen mukaisesti. Sen sijaan manuaalisesti sijoitetut kurssit vaativat aina tarkastamisen.
- *Arvioida opiskelijan HOPS.* Opintosihteeri voi sanallisen arvioinnin lisäksi arvioida suunnitelmaa tutkintotasolta aina yksittäisen kurssin tasolle asti. Jokaisen opintosuunnitelmaan sisällytettyyn opintoryhmään ja opintojaksoon sisältyy tieto sen tilasta. Tämän tiedon perusteella voidaan erotella hyväksytyt, hylätyt tai vielä arvioimatta olevat opintoryhmät ja -jaksot. Tästä saatava hyöty ilmenee, kun hylätty opintosuunnitelma palautetaan takaisin opiskelijalle korjattavaksi. Opiskelija korjaa suunnitelmastaan virheellisiksi merkityt kohdat ja lähettää suunnitelman arvioitavaksi uudelleen. Tässä vaiheessa opintosuunnittelijan ei enää erikseen tarvitse tarkastaa koko opintosuunnitelmaa ja jo aiemmin hyväksyttyjä osia, vaan ainoastaan aiemmin hylätyt tai muut hyväksynnän jälkeen muokatut osat. Näin opintosuunnitelman tarkastaminen myöhemmillä arviointikierroksilla nopeutuu.

### 4.3.3 Ylläpitäjä

Järjestelmän käytettävyys vaatii keskitettyä ylläpitoa ja riittävän säännöllistä päivittämistä. Yksi ylläpitäjälle kuuluvista tehtävistä on vastata järjestelmän käyttöoikeuksista. Koska käyttäjien autentikointi tapahtuu LTY:n ympäristössä keskitetysti,

oletusarvoisesti kaikilla opiskelijoilla on pääsy järjestelmään. Sen sijaan opintosihteereille ja ylläpitäjille on luotava paikalliset käyttäjätilit, joihin sidotaan heidän vastuulleen kuuluvat tehtävät.

Järjestelmän ylläpitäjän päätehtäviin kuuluu:

- *Hallita järjestelmän käyttöoikeuksia.* Vain rajatulla ryhmällä tulee olla mahdollisuus tarkastella yksittäisen opiskelijan HOPSia. Ylläpidon tehtävänä on määrittää ja tarvittaessa muokata opintosihteereiden ja opiskelijoiden välisiä oikeuksia.
- *Hallita opintojaksoja.* Yliopiston kurssitarjonta muuttuu vuosittain. Uudet kurssit tulee lisätä järjestelmän tietokantaan, jotta niitä voidaan täysipainoisesti hyödyntää suunnittelussa.
- *Hallita tutkintoja, opintokokonaisuuksia sekä niihin liittyviä opintojaksoja.* Tutkintojen sisällöt muuttuvat vuosittain. Niinpä järjestelmään on mallinnettava uudet tutkintorakenteet, jotta niitä voidaan käyttää opintojensuunnittelun tukena.
- *Poistaa opiskelijoiden tiedot järjestelmästä.* Valmistuneiden tai muuten lopettaneiden opiskelijoiden tiedot tulee poistaa järjestelmän tietokannasta.

Ylläpitäjän tehtäviin voi myös kuulua vastuu tutkintorakenteiden päivittämisestä. Tämä tehtävä on myös mahdollista sisällyttää opintosihteerien vastuulle. Keskitetty tutkintorakenteiden hallinta on kuitenkin tehokkain vaihtoehto, mikäli se on saatavilla olevien resurssien puitteissa mahdollista toteuttaa. Koska rakenteiden syöttäminen on yksittäinen, kerran vuodessa suoritettava tapahtuma, on käyttö todennäköisesti omaksuttava jokaisella kerralla ainakin osittain uudelleen. Kun yksi henkilö kerrallaan vastaa tutkintorakenteiden ylläpidosta, ei jokaisen opinto-ohjaajan tarvitse erikseen opetella järjestelmän käyttöä, vaan he voivat tarvittaessa avustaa osastokohtaisissa tutkintorakenteeseen liittyvissä kysymyksissä.

## 4.4 Käytettävyys

Kuten roolit, myös järjestelmän käyttötapa vaihtelee eri kohderyhmien kesken. Opiskelijoiden käyttö saattaa olla hyvinkin vaihtelevaa; toiset päivittävät suunnitelmaansa aktiivisesti, toiset harvemmin tai vain tarvittaessa. Opintosihteerit käyttävät järjestelmää aktiivisimmin, sillä heidän vastuualueelleen kuuluu suurehko määrä opis-

kelijoita, joiden suunnitelmia he tarkastavat ja kommentoivat. Ylläpitäjän kannalta järjestelmän käytön voidaan olettaa olevan varsin kausittaista. Niinpä myös käytettävyydelle ja käyttöliittymille asetettavat vaatimukset ja tavoitteet poikkeavat toisistaan.

# Luku 5

# Järjestelmän toteutus

Suunnitteluvaiheen jälkeen aloitettiin järjestelmän tekninen toteutus. Toteutusvaihe jaettiin erillisiin tehtäväkokonaisuuksiin. Pääosan huomiosta vei sovelluksen toiminnallisuuden ja järjestelmäarkkitehtuurin toteuttaminen. Toimintojen valmistuttua siirryttiin käyttöliittymän toteuttamiseen, jonka avulla toiminnot sidottiin toimivaksi kokonaisuudeksi.

Toteutusvaiheessa pyrittiin kiinnittämään varsin paljon huomiota sovelluksen toimintavarmuuteen. Tämä käsittää myös käyttäjien tekemien virheiden käsittelyn sekä niistä mahdollisesti aiheutuvien ongelmien estämisen. Näihin seikkoihin tutustutaan tarkemmin tässä luvussa.

Toteutusvaiheen alussa valittiin käytettävät tekniset ratkaisut, joiden avulla suunniteltu järjestelmä kyettiin toteuttamaan. Sovelluksen järjestelmäalustaksi valittiin LAMP-pohjainen ratkaisu (Linux, Apache, MySQL ja PHP), joka on tällä hetkellä varsin suosittu alusta WWW-pohjaisille sovelluksille. Kaikki järjestelmän käyttöön tarvittavat ohjelmistot ovat vapaasti levitettäviä.

## 5.1 Toteutuksessa käytetyt tekniikat ja sovellukset

WWW-sovelluksen ja sen käyttäjän välisen informaation esittämiseen käytettäviä tekniikoita valittaessa on otettava huomioon niiden mahdollisimman laaja yhteensopivuus ja toimintavarmuus erilaisilla sovellusalustoilla ja selaimilla. Niinpä sovellusympäristön tulee noudattaa yleisesti hyväksyttyjä standardeja, eikä selainkohtaisia laajennuksia ole suositeltavaa käyttää. Tästä johtuen sovelluksen toiminnoissa ei tulla käyttämään esimerkiksi JavaScript:iä. Sovelluksen käyttö tulee olla mahdollis-

ta myös sellaisessa tapauksessa, jossa käyttäjän ympäristö ei tue jotain käytettyä tekniikkaa, kuten evästeitä tai tyylitiedostoja.

Sovelluksen toteutukseen käytettäviä tekniikoita, PHP-skriptikieltä (PHP: Hypertext Preprocessor) ja SQL-pohjaista (Structured Query Language) tietokantaa, valittaessa otettiin huomioon toteutettavan sovelluksen toiminnallisuuden asettamat vaatimukset sekä mahdollisimman hyvä yhteensopivuus erilaisten palvelualustojen välillä. Toteutuksessa käytettävä ympäristö perustuu yleisiin ja tehokkaisiin työkaluihin WWW-pohjaisia sovelluksia toteutettaessa.

### 5.1.1 HTML

HTML (Hypertext Markup Language) on WWW-sivun sisällön rakenteen kuvaamiseen käytettävä kuvauskieli. Kieli perustuu tagien käyttöön, joiden avulla muodostetaan rakenteellisia elementtejä. Tagien avulla ei niinkään kuvata esitettävää informaatiota ja sen sisältöä, vaan sen esitystapaa. WWW-selaimet esittävät sivun sisältämän informaation rakenteellisena dokumenttina.

Sovelluksen toteuttamisvaiheessa tulee ottaa huomioon, että kaikki tuotettu HTML-koodi noudattaa sille asetettua standardia, joka on määritetty W3C:n (World Wide Web Consortium) [1] toimesta. Sen tarkka noudattaminen edesauttaa palvelun yhteensopivuutta erilaisten selainten ja niiden eri versioiden välillä. Toteutuksessa minkään osan ei tule perustua yksittäisen selaimen tukemien toimintojen käyttöön.

### 5.1.2 CSS tyylitiedostot

Ulkoasun asettelun, fonttien ja sijoittelun määrittämiseen voidaan käyttää CSS 2.1-määrittelyn [29] mukaisia tyylitiedostoja. Tyylitiedostojen avulla jokaiselle HTML-elementille kyetään määrittämään sekä yleiset että tarkemmin rajaavat ryhmäkohtaiset asetukset CLASS- ja ID-ominaisuuksien avulla. Tyylitiedostoja käyttämällä sivulle saadaan luotua yhtenäinen ja helposti hallittavissa oleva ulkoasu. Mikäli sivuston tyyliä halutaan muuttaa, onnistuu se muokkaamalla sivuston käyttämää tyylitiedostoa, eikä jokaista näkymää erikseen muokkaamalla.

Tyylitiedostojen käyttö mahdollistaa myös erilaisten näkymien toteuttamisen eri medioille. Käyttöliittymästä voidaan toteuttaa toisistaan eriävät tyylit kahdelle eri medialle, kuten näytölle sekä tulosteelle. Tulosteesta voidaan karsia pois varsinaista sisältöä lukuunottamatta kaikki ylimääräinen informaatio, kuten navigointi- ja otsik-

[1] `http://www.w3c.org/`

kosolut. Näin selaimen näkymästä voidaan automaattisesti luoda myös tulostettava versio, eikä sivusta tarvitse tuottaa erillistä versiota tulostusta silmälläpitäen.

### 5.1.3 Keksit

Keksit, joista käytetään myös nimeä evästeet, ovat WWW-sovelluksen asiakkaan päätteelle tallentamia tiedostoja, jotka sisältävät palvelun määrittämää sen toiminnan kannalta tarpeellista tietoa. Keksiin sisällytettävä tieto lähetetään HTTP:n (Hypertext Transfer Protocol) otsikkotiedoissa, joista käytettävä WWW-selain erottelee sen. Mikäli selain on aiemmin tallentanut kyseisen palvelimeen liittyvän keksin, lähetetään myös siihen sidotun evästeen tiedot automaattisesti jokaisen pyynnön yhteydessä. [30]

Keksejä hallitaan WWW-sovelluksesta käsin. Niiden avulla käyttäjästä voidaan tallentaa tietoa, jota voidaan hyödyntää joko saman tai myöhemmän istunnon aikana. Usein evästeeseen tallennettu tieto on istuntokohtaista, kuten verkkokaupan ostoskorin sisältö tai istunnossa käyttäjän tunnistamisessa hyödynnettävä tunniste. Kaikki selaimet eivät välttämättä tue evästeiden käyttöä, tai käyttäjä itse on estänyt niiden käytön, joten toteutuksessa on otettava huomioon mahdollisuus tällaisen tapauksen varalle.

### 5.1.4 XML ja XML-skeema

XML (Extensible Markup Language) [31] on SGML:ään (Standard Generalized Markup Language) perustuva kuvauskieli. Tavoitteena on ollut muodostaa SGML:ää yksinkertaisempi kuvaustapa, joka kuitenkin mahdollistaa HTML:stä poiketen itse asiasisällön, ei pelkän esitystavan, kuvaamisen. XML:n käyttö perustuu metatietoa määrittävien tunnisteiden käyttöön, joiden avulla tiedoston sisältämät tiedot pyritään määrittelemään.

Kuten tietokantojen ja olioiden yhteydessä, myös XML:n rakenteelle, tietotyypeille ja käsittelylle täytyy määritellä säännöt. XML-dokumentin rakenne voidaan määritellä XML-skeeman avulla. Jo ennen XML-skeemoja rakenne pystyttiin kuvaamaan DTD:n (Document Type Definition) avulla, joka ei kuitenkaan sisältänyt kaikkia skeemojen suomia mahdollisuuksia. Lisäksi XML-skeemasta poiketen DTD:n kielioppi ei perustu XML:ään.

### 5.1.5 PHP

PHP on suosittu, pääosin WWW-ohjelmoinnissa käytettävä, palvelimella tulkattava skriptikieli. PHP-skripti suoritetaan palvelimella ja sen tuottama informaatio, kuten HTML-dokumentti, esitetään käyttäjälle. [30] Komennot voidaan halutessa sisällyttää suoraan HTML-kuvauskielen sekaan. PHP:n tulkkaaminen on mahdollista erilaisilla käyttöjärjestelmäalustoilla.

PHP sisältää rajapinnat useisiin eri tietokantoihin, joka mahdollistaa dynaamisesti toimivien sivujen toteutuksen. Se sisältää rajapinnan tässä diplomityössä toteutettavassa järjestelmässä käytettävään MySQL-tietokantaan sekä tukee myös ODBC-standardia (Open DataBase Connectivity). PHP:n viimeisin versio on 5, joka mahdollistaa oliopohjaisen ohjelmoinnin. Toteutettava sovellus perustuu PHP:n versiossa 4 käytössä oleviin toimintoihin, eikä hyödynnä oliopohjaisuutta.

### 5.1.6 SQL

SQL on tietokannan käsittelyyn käytettävä kyselykieli. SQL on sekä ANSI:n (American National Standards Institute) että ISO:n (International Organization for Standardization) määrittelemä standardi, jota useimmat tietokantasovellukset tukevat. Tietokantaan suoritettavien komentojen, kyselyjen, avulla voidaan hakea, lisätä, päivittää ja poistaa tietoja. Suuri osa dynaamisista WWW-sivuista perustuu SQL-tietokannan käyttöön tietovarastona.

Sovelluksessa käytettäväksi relaatiotietokantapalvelimeksi valittiin MySQL[2]. Se on avoimeen lähdekoodiin perustuva tietokantaratkaisu rakenteellisen tiedon tallentamiseen ja kuvaamiseen. Koska toteutettavan sovelluksen toiminta perustuu hyvin pitkälle tietokannassa kuvatun informaation rakenteellisuuteen, joudutaan myös käytettävälle palvelimelle asettamaan vaatimuksia. MySQL:n toiminnallisuus vastasi tietokantaratkaisulle asetettuja vaatimuksia ja yhtenä suosituimmista ja jatkuvasti kehitettävänä se tarjoaa myös riittävän toiminnallisen luotettavuuden ja jatkuvuuden.

## 5.2 Sovelluksen käyttöoikeuksien hallinta

Sovellukseen toteutettavan tietoturvan pääperiaatteena on, että käyttäjällä on käyttö- ja katseluoikeudet vain siihen tietoon, johon hänellä on oikeudet. Tästä johtuen käyttäjät autentikoidaan sisäänkirjautumisen yhteydessä. Onnistuneen autentikoinnin

[2] `http://www.mysql.com/`

jälkeen käyttäjälle luodaan henkilökohtainen istunto, joka tunnistetaan istuntoavaimen perusteella. Sovelluksen toiminnan kannalta ehdoton edellytys on, että käyttöoikeuksien sekä istuntojen hallinta on luotettavaa ja sovelluksen tietoturvataso on riittävä.

Opiskelijoiden autentikointi pyritään pääasiallisesti suorittamaan RADIUS-palvelun avulla. RADIUS-tunnistusta käytettäessä annettu käyttäjätunnus-salasanapari lähetetään yhteisen avaimen avulla salattuna viestinä RADIUS-palvelimelle, jossa autentikointi suoritetaan. Tämän jälkeen palvelin lähettää sovellukselle takaisin tiedon, onko käyttäjän tunnistaminen onnistunut.

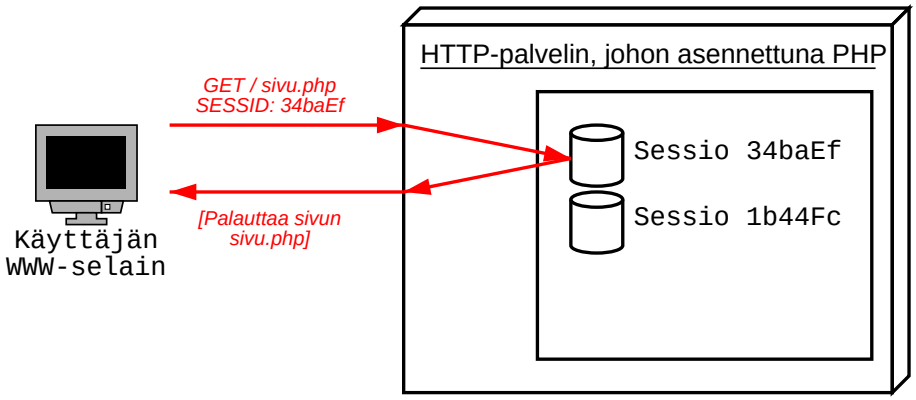
Toinen mahdollisuus opiskelijoiden tunnistamiseen on käyttää paikallisesti tallennettuja käyttäjätietoja. Tällöin tunnus on tallennettu lokaalisti järjestelmän omaan tietokantaan. Lisäksi kaikkien palvelun ylläpitäjien autentikointi tapahtuu lokaalien tilien avulla.

Tietokantaan tallennetaan käyttäjätunnus sekä salasana, joiden avulla käyttäjien tunnistus tapahtuu. Salasanaa ei tallenneta tietokantaan selkokielisenä, vaan siitä tallennetaan MD5-algoritmilla muodostettu 128-bittinen tiiviste, hash. Algoritmin avulla alkuperäisestä salasanasta luodun yksisuuntaisen tiivisteen tallentaminen varsinaisen salasanan sijaan takaa sen, etteivät käyttäjän salasanat vääriin käsiin joutuessaan ole saatavilla selkokielisinä. Sisäänkirjautuvaa, paikallisen käyttäjätilin omaavaa, käyttäjää tunnistettaessa hänen salasanastaan lasketaan jälleen hash-arvo ja verrataan sitä tietokantaan tallennettuun arvoon.

Onnistuneen autentikoinnin jälkeen käyttäjälle luodaan henkilökohtainen istunto, jota myöhemmin käytetään palvelimelle lähetettävien pyyntöjen autorisointiin. WWW-palvelimien ja selainten välisessä liikenteessä käytettävä HTTP on tilaton protokolla eli se ei tallenna automaattisesti tilatietoa pyyntöjen välillä. Niinpä yhteys asiakkaan ja palvelimen välillä muodostetaan jokaisen pyynnön yhteydessä erikseen. Istuntopohjaisessa palvelussa tietyn tilatiedon ylläpito on kuitenkin välttämätöntä.

Istunnot tunnistetaan yksilöllisen avaimen perusteella. Kullekin istunnolle erikseen luotava avain on 25 merkkiä pitkä, joka tekee sen puhtaasta arvaamisesta käytännössä lähes mahdotonta. Kuvassa 5.1 on esitetty istuntojen toiminnan periaate. Asiakaspäässä istuntoavain tallennetaan selaimen keksiin ja se lähetetään palvelimelle jokaisen pyynnön yhteydessä. Palvelimella istuntoavaimet ovat tallennettuna tietokantaan sidottuna istunnon omistajan käyttäjätunnukseen. Jokaisen palvelimelle lähetetyn pyynnön yhteydessä tarkastetaan, että vastaanotettua avainta vastaava istunto on olemassa, eikä se ole vanhentunut. Aikaleima on tallennettu tietokantaan

yhdessä istuntoavaimen ja käyttäjätunnuksen kanssa ja kukin istunto vanhenee, mikäli sitä ei ole käytetty määritetyn ajan kuluessa. Istunnon pituudeksi toteutetussa palvelussa on asetettu 30 minuuttia.




Kuva 5.1: Web-selain lähettää istuntoavaimen palvelimelle jokaisen pyynnön yhteydessä. [32]

## 5.3 Opintosuoritusten automaattinen sijoittaminen HOPSiin

Opintosuunnitelman luomista pyritään helpottamaan mahdollistamalla valmiiksi suoritettujen opintojen automaattinen sijoittaminen laadittuun tutkintorakenteeseen. Toiminto on hyödyllinen pääasiassa niille opiskelijoille, joilla on lukuisia suunnitelmaan sijoittamattomia suorituksia. Kun opiskelija ensimmäistä kertaa luo tai myöhemmin päivittää HOPSiaan, toiminnon avulla pystytään vähentämään opiskelijalta vaadittavaa mekaanista työtä.

Toimintoa varten toteutettiin algoritmi, joka käy yksi kerrallaan läpi jokaisen tutkintorakenteeseen valitun opintoryhmän rakenteen määräämässä järjestyksessä perusopinnoista alkaen. Ensin etsittäväksi määritetään kaikki alkuperäisessä tutkintorakenteessa kyseiseen ryhmään määritetyt opintojaksot. Tämän jälkeen tästä ryhmästä poistetaan kaikki ne, jotka opiskelijalla jo on sijoitettuna kyseisessä ryhmässä. Lisäksi alkuperäisistä hakuehdoista poistetaan myös ne opintojaksot, joita vastaava tai korvaava kurssi on sijoitettuna ryhmässä. Tämän jälkeen saaduilla hakuehdoilla suoritetaan vertailu opiskelijan niiden kurssien kanssa, joita ei vielä ole sijoitettu mihinkään tutkinnon opintoryhmään. Tämän jälkeen kaikki hakuehdoilla löytyneet opintojaksot siirretään automaattisesti kyseiseen ryhmään. Sama toiminto suorite-

taan jokaiselle opintoryhmälle erikseen.

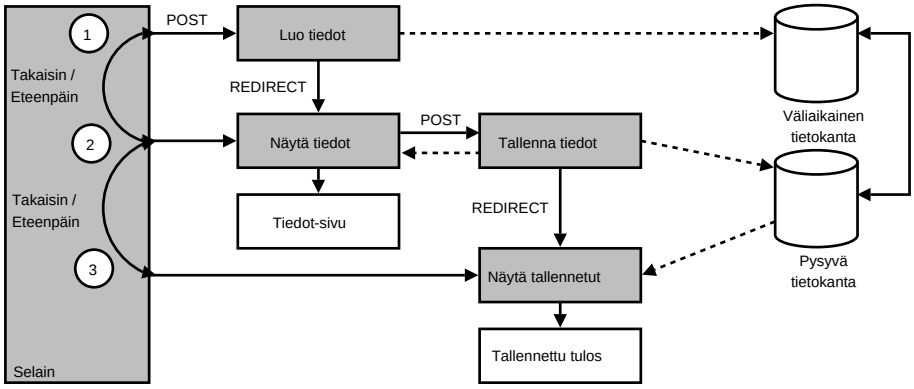
## 5.4 Kahdenkertaisen lähettämisen ongelma

HTTP:n GET-metodilla voidaan URI:n (Uniform Resource Identifier) avulla viitata suoraan toimintoon ja resurssiin. Niinpä sitä tulisi käyttää ainoastaan, mikäli pyynnöstä suoritetaan ns. turvallinen operaatio, kuten yksinkertainen haku tietokannasta. POST-metodi on soveltuva toisentyyppisiin toimintoihin, joissa käyttäjän lähettämä pyyntö mahdollisesti vaihtaa resurssin tilaa. [33] Esimerkkejä tällaisista toiminnoista ovat kaikki tietokantaan suoritettavat muutokset.

Koska HTTP on tilaton protokolla, ei käyttäjän tilaa voida seurata automaattisesti ilman sovellukseen lisättyä erillistä toteutusta. Tämä aiheuttaa ongelmia tilanteissa, joissa käyttäjä lähettää saman pyynnön, joka muuttaa sovelluksen tilaa tai tietoja, uudelleen palvelimelle. Tällainen tilanne tulee useimmin vastaan, kun käyttäjä joko päivittää jo ladatun sivun, palaa takaisin aiemmin ladatulle sivulle ja lataa sen uudelleen tai palaa takaisin lomakkeeseen ja lähettää täyttämänsä tiedot uudelleen. Niinpä tämä ongelma on pyrittävä estämään, jotta tietokannan eheys kyetään säilyttämään.

Michael Jouravlev on artikkelissaan [34] esittänyt tavan kahdenkertaisen lähettämisen ongelman, ns. "Double Submit Problem", ratkaisemiseksi. Ratkaisu perustuu uudelleenohjaukseen ja väliaikaisen tietokannan käyttöön ja sen toimintaperiaate on esitetty kuvassa 5.2. POST-metodia käyttäen lähetetyt lomakkeen tiedot tallennetaan väliaikaiseen tietokantaan, jonka jälkeen käyttäjä ohjataan automaattisesti uudelle sivulle GET-metodin avulla. Vasta tällä sivulla käyttäjälle lähetetään suorituksen aiheuttama palaute. Koska kyseisen ratkaisun toteuttaminen vaatii suhteellisen paljon huomioimista toteutuksen rakenteessa, vaatii ylimääräisen tietovaraston luomisen ja käyttämisen sekä jättää osan toiminnan vastuusta selaimelle, sen käyttö ei tullut kysymykseen.

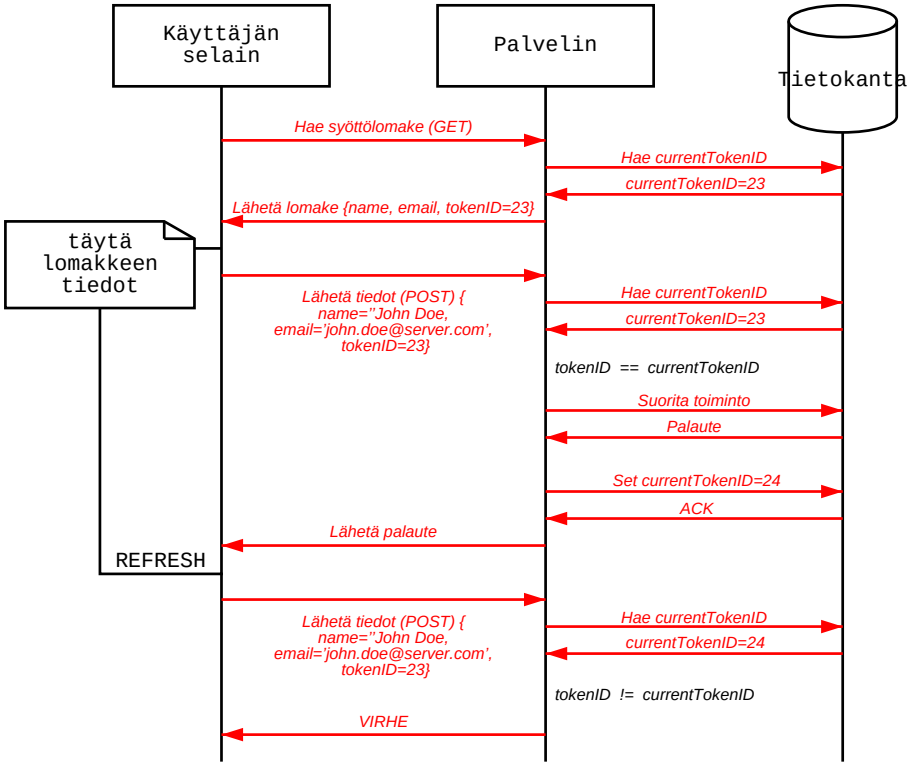
Koska kahdenkertaisen lähettämisen ongelma oli joka tapauksessa ratkaistava, oli sovellukseen kehitettävä kevyempi toteutus. Toteutus perustuu vanhentuvan tunnisteen käyttöön, joka sisällytetään jokaiseen lomakkeeseen. Tämä tunniste on aina sidottu aktiivisena olevaan istuntoon ja se lähetetään palvelimelle muiden tietojen yhteydessä. Kun tietokannan sisältöön vaikuttava pyyntö lähetetään palvelimelle, ensin tarkastetaan, että lähetettyä tunnistetta ei ole vielä käytetty. Mikäli se on käytetty ja tunniste on vanhentunut, ei pyydettyä toimintoa suoriteta, vaan tulostetaan




Kuva 5.2: Uudelleenohjaus POST-metodin jälkeen. [34]

ainoastaan toimintoa seuraava palaute. Mikäli tunniste on käyttämätön, suoritetaan toiminto ja samalla luodaan kyseiselle istunnolle uusi tunniste, joka sisällytetään myöhemmin palvelimelta lähetettyihin lomakkeisiin. Mikäli tässä vaiheessa käyttäjä yrittää suorittaa saman toiminnon uudelleen aiemmin kuvatulla tavalla, lähetetään palvelimelle automaattisesti lomakkeen tietojen mukana vanhentunut tunniste, joka ei enää kelpaa. Kaikki palvelun sisältämät lomakkeet, jotka aiheuttavat muutoksia tietokantaan, on suojattu kyseisellä metodilla.

Kuvassa 5.3 on esitetty esimerkkitapaus lomakkeen lähettämisestä. Ensin käyttäjä lataa lomakkeen, jonka tiedot täytetään. Hän lähettää lomakkeen palvelimelle, jossa tarkastetaan ja havaitaan tunnisteen käyttökelpoisuus. Käyttäjän pyytämä päivitys tietoihin suoritetaan ja hänelle lähetetään palaute toiminnosta. Tämän jälkeen käyttäjä hakee saman sivun uudelleen selaimen “Päivitä”-toiminnon avulla. Koska sivu on palaute aiemmin POST-pyynnöllä lähetettyyn resurssiin, yritetään sama toiminto suorittaa uudelleen samoilla arvoilla. Palvelimella havaitaan tässä vaiheessa tunnisteen olevan vanhentunut, joten pyydettyä muutosta tietokantaan ei suoriteta.




Kuva 5.3: Uudelleenlähetyksen eston toimintakaavio.

## 5.5 Käyttöliittymän ulkoasu

Käyttöliittymän rakenne on määritetty erillisessä tiedostossa, jonka avulla sisältö on jaettu erillisiin osiin. Yläosa, navigointi ja alaosa on kaikille yhteinen, ainoastaan sisältö vaihtuu toiminnosta riippuen. Hallinnan keskittäminen mahdollistaa sivujen välisen yhdenmukaisuuden sekä tarvittaessa rakenteen muokkaamisen hallitusti. Kuvassa 5.4 on esitetty kuvankaappaus opiskelijoiden käyttöliittymästä, joka perustuu navigointikeskeiseen malliin.

Tyylitiedostojen avulla sovellukseen toteutettiin tuki kahdelle erilliselle medialle. Mikäli käyttäjä tulostaa näkymän, ainoastaan sisältö tulostetaan. Käytännössä tämä tapahtuu määrittämällä muut näkymän solut näkymättömiksi tulostusmedialle. Näin erillistä näkymää tulostetta varten ei tarvita.

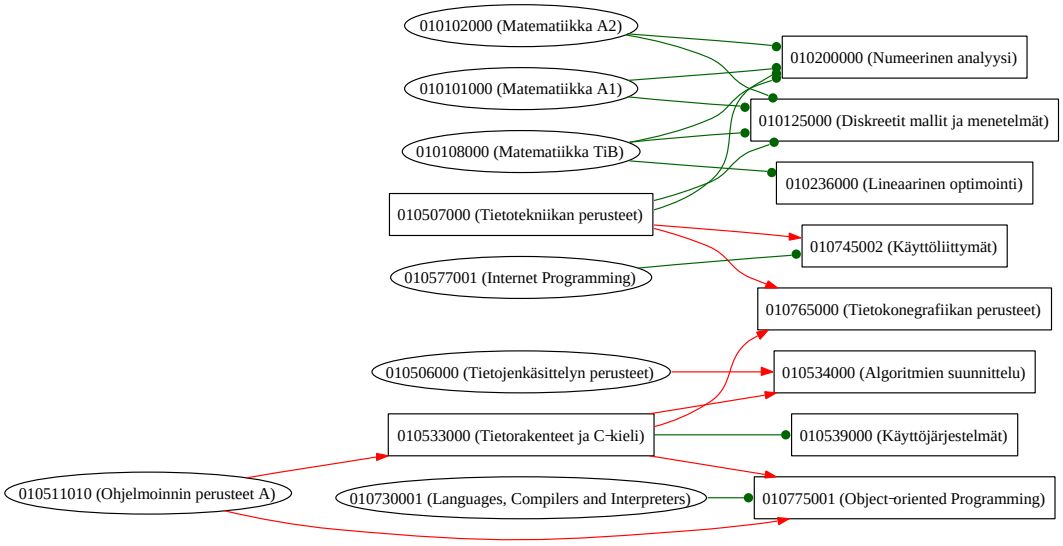

Kuva 5.4: Opiskelijoiden käyttöliittymän ulkoasu.

## 5.6 Esitietovaatimusten graafinen mallinnus

Järjestelmää toteutettaessa tuli esiin opintojaksojen esitietovaatimukset. Esitiedot voidaan ottaa huomioon kursseja valittaessa niin, että järjestelmä rakentaa automaattisesti esitietoihin perustuvan puurakenteisen mallin opintosuorituksista. Mallista havaitaan helposti, mikäli jonkun suunnitelmaan valitun opintojakson esitietona vaadittu suoritus puuttuu. Käytännössä havainnollistaminen toteutetaan siten, että tiedoista muodostetaan kuvatiedosto, jossa opintojaksojen väliset relaatiot on kuvattu graafina.

Graafin piirtäminen toteutettiin Graphviz-ohjelmistolla[3], jota levitetään CPL-lisenssin (Common Public License) alla. Ensin opiskelijan valitsemista kursseista ja niiden tietokannassa määritetyistä esitietovaatimuksista luodaan tekstipohjainen DOT-tiedosto, jossa kuvaan sisällytettävät komponentit sekä niiden väliset suhteet. Kuvaustiedoston rakenne on esitetty liitteessä 3. Tämän jälkeen tiedostosta luodaan Graphviz-ohjelmiston avulla PNG-formaatissa (Portable Network Graphics) oleva kuva, jossa havainnollistetaan kurssien keskinäiset riippuvuussuhteet. Esimerkki näin luodusta kurssigraafista on kuvassa 5.5. Opiskelijan suunnitelmaan sisällytetyt kurssit on sijoitettu suorakulmaisiin laatikoihin ja puuttuvat kurssit ovaaleihin. Pakollinen esitietovaatimus on kuvassa yhdistetty opintojaksoon nuolella, suositeltava esitieto ympyrällä. Rakenteen avulla esitietovaatimusten ja -suositusten täyttyminen on helppo havainnoida ja tarvittaessa lisätä puuttuvat kurssit suunnitelmaan.

[3] `http://www.graphviz.org/`




Kuva 5.5: Osa Graphviz:n avulla luotua graafia opintojaksojen esitietovaatimuksista.

Lopulliseen järjestelmään ei toimintoa ole toistaiseksi kuitenkaan lisätty. Esitietojen rooli tämän sovelluksen puitteissa ei ole kovin tärkeä, mutta tulevaisuuden tilannetta ja mahdollista tarvetta on vaikea arvioida. Tämän toiminnallisuuden hyödyntäminen vaatii lisäksi Graphviz-ohjelmiston asentamisen palvelimelle.

## 5.7 Rajapinnat

Järjestelmän tarvitsemaa informaatiota on tallennettuna useaan paikkaan. Opiskelijan opintosuoritukset on LTY:ssa tallennettu Oodin tietokantaan. Jotta tietoa voitaisiin siirtää hallitusti eri järjestelmien välillä, tarvitaan yhteiset rajapinnat ja yhteinen esitystapa siirrettävälle tiedolle.

Kun opiskelija on luonut oman opintosuunnitelmansa, on hänellä mahdollisuus kopioida se itselleen HTML-dokumenttina ja tulostaa se. Lisäksi opintosuunnitelman voi saada järjestelmästä ulos XML-pohjaisena tiedostona.

### 5.7.1 RADIUS

WebTUTOR-sovellukseen toteutettiin mahdollisuus tunnistaa käyttäjät RADIUS-palvelun avulla. Yliopiston tarjoama keskitetty käyttäjähallinta helpottaa järjestelmän hallittavuutta, sillä järjestelmän ylläpidon ei tarvitse erikseen lisätä jokaista

käyttäjää. Sitävastoin kaikki henkilöt, jotka voidaan autentikoida RADIUS-palvelun avulla, voivat automaattisesti kirjautua myös WebTUTORiin.

Sovellus toimii tunnistusprosessissa asiakkaana, joka lähettää palvelimelle käyttäjätunnuksen ja salasana jaetun salaisuuden avulla salattuna. Tämän jälkeen RADIUS-palvelin purkaa vastaanotetun viestin ja tarkastaa käyttäjätunnuksen ja salasanan oikeellisuuden tietokannastaan. Tämän jälkeen asiakkaalle lähetetään viesti, jossa ilmoitetaan autentikoinnin onnistumisesta, epäonnistumisesta tai mahdollisesta virheestä.

Sovelluksessa RADIUS-tunnistukseen käytetään Edwin Groothuisin luomaa PHP-RADIUS-kirjastoa. Se on BSD-lisenssin (Berkeley Software Distribution) alainen ja sallii ohjelman käyttämisen, muokkaamisen, kopioinnin sekä levittämisen. Kirjastoa jouduttiin muokkaamaan poikkeus- ja virhetilanteiden käsittelyn osalta niin, että mahdollinen poikkeustilanne ei keskeytä toimintoa kutsuneen skriptin suoritusta. Tällainen tilanne voi johtua tietoverkkohäiriöstä, autentikoinnin tai yhteydenmuodostuksen epäonnistumisesta tai mahdollisesti myös sovelluksesta riippumattomasta tekijästä.

### 5.7.2 Web Services

Tutkintorakenne- ja kurssitieto voidaan saada järjestelmästä joko sovelluksen oman WWW-käyttöliittymän avulla tai XML-tiedostona. Kurssitiedon muoto perustuu Tieteen tietotekniikan keskuksen laatimaan XML-skeemaan [35]. Erilaisia raportteja voidaan käsitellä myös joko käyttöliittymän kautta tai XML-pohjaisina tiedostoina. Järjestelmän käyttäjillä on mahdollisuus hyödyntää rajapintoja tarpeidensa mukaan. Toteutusvaiheessa on huomioitu järjestelmän toiminta itsenäisenä sovelluksena ja sen riippumattomuus muista käytössä olevista ratkaisuista. Opiskelijat voivat ladata oman opintosuoritusotteensa sovellukseen, jolloin siitä generoidaan suorituslista järjestelmän tietokantaan.

Web Services–määrittely määrittelee standardin tiedonsiirtotavan toisistaan erillisten, mahdollisesti toisistaan poikkeavissa ympäristöissä toimivien sovellusten väliseen tiedonsiirtoon. Web Service–ratkaisua käyttävä sovellus sisältää rajapinnan, joka on kuvattu konepohjaisen käsittelyn mahdollistavassa formaatissa (WSDL-formaatti). Muut järjestelmät kommunikoivat määritettyjä sääntöjä noudattaen Web Service-rajapinnan kanssa SOAP-viestejä (Simple Object Access Protocol) käyttäen, joita tyypillisesti kuljetetaan HTTP-protokollan avulla XML-muodossa olevana informaationa. [36]

Järjestelmän toteutuksen yhteydessä testattiin Web Services –rajapinnan toimivuutta ja mahdollista myöhempää hyödyntämistä tiedonsiirrossa NuSOAP-ohjelmistokirjaston[4] avulla. Koska tällä hetkellä WebTUTORiin kytköksissä olevat järjestelmät eivät tue tätä rajapintaa, emme katsoneet tarpeelliseksi sisällyttää tätä tekniikkaa lopulliseen versioon. Koska kuitenkin erilaisia raportteja on saatavilla XML-muodossa, ei tarpeen tullen palvelun lisääminen vaadi suuria muutoksia jo olemassaolevaan järjestelmään. Tällöin esimerkiksi kurssitiedon, tutkintorakenteen tai yksittäisen opiskelijan opintosuunnitelman siirtäminen toiseen järjestelmään tapahtuu vaivattomasti ja käyttäjälle mahdollisimman näkymättömästi.

### 5.7.3 Tuotantopalvelimen ja kehityspalvelimen välinen rajapinta

Kun järjestelmän tietokantaa päivitetään uusia tutkintorakenteita luomalla, ei muutoksia kannata toteuttaa tuotantopalvelimen varsinaiseen tietokantaan suoraan. Suositeltavaa on, että tiedot tallennetaan ensin kehityspalvelimelle, josta ne myöhemmin siirretään varsinaisen tuotantopalvelimen käyttöön. Toinen mahdollisuus on hyödyntää tuotantopalvelimelle luotua erillistä tietokantaa. Tällöin ennen julkista käyttöönottoa voidaan suorittaa testaus, jotta mahdolliset ongelmakohdat havaitaan ja samalla voidaan estää opiskelijoilta pääsy keskeneräisiin tutkintorakennepohjiin. Tutkintorakenteen luominen tuotantopalvelimelle sijoitettuihin väliaikaistauluihin on suositeltavaa ainoastaan siinä tapauksessa, että erillistä kehityspalvelinta ei ole käytettävissä.

Tiedon kopioiminen kehityspalvelimelta tuotantopalvelimen käyttöön tapahtuu suoran tietokantayhteyden avulla. Suoritukseen oikeudet omaava henkilö näkee listan kehitysvaiheessa olevista tutkintorakenteista. Valittuaan halutun tutkintorakenteen siirrettäväksi, valittu tutkintorakenne kopioidaan automaattisesti tuotantopalvelimelle. Kopio tutkintorakenteesta säilytetään kehityspalvelimella.

Syy suoraan tietokantayhteyteen pohjautuvan rajapinnan käyttöön on se, että kyseessä on järjestelmän sisäinen tiedonsiirto. Tällä tapaa siirto on huomattavasti nopeampaa ja turvallisempaa kuin esimerkiksi Web Services -rajapintaa käyttämällä. Sovellusta voidaan tarvittaessa muokata tukemaan toista tiedonsiirtotapaa myös palvelinten välillä tapahtuvassa tietoliikenteessä.

[4] `http://sourceforge.net/projects/nusoap/`

# Luku 6

# Testaus ja käyttöönotto

Testaus on välttämätön vaihe sovelluksen laadun ja toiminnan varmistamisessa. Testaamista suoritettiin jatkuvasti sovelluksen toteuttamisen eri vaiheissa, yksikkötestaamisesta aina järjestelmätasolle asti. Jokaisessa vaiheessa suoritetun testaamisen avulla pyrittiin löytämään ohjelmistossa mahdollisesti piilevät virheet mahdollisimman nopeasti. Samalla tarkasteltiin järjestelmän toimintaa normaalista poikkeavissa tilanteissa.

Varsinaisesta testaamisesta voidaan erottaa omaksi osakseen käytettävyystestaus ja suorituskyvyn analysointi. Ennen sovelluksen julkista käyttöönottoa analysoitiin sen toimintaa ja sen käytössä mahdollisesti eteen tulevia ongelmia. Näissä vaiheissa pyritään havaitsemaan käytettävyyden kannalta esiintyviä ongelmia, ei niinkään ohjelmallisia virheitä.

## 6.1 Käytettävyyspalautteen arviointi

Järjestelmälle suoritettiin kaksi erillistä käytettävyystestiä. Ensimmäinen vaihe suoritettiin järjestelmän prototyypin valmistuttua. Viisi henkilöä osallistui sovelluksen toiminnan arviointiin ja jokaiselta pyydettiin palautetta järjestelmän toiminnallisuudesta, toimintojen tarpeellisuudesta sekä mahdollisesti esille tulleista puutteista ja ongelmista. Saatu palaute koski pääosin järjestelmän käyttöliittymään, navigointiin ja toiminnallisuuteen liittyviä seikkoja, joita pyrittiin tarpeellisilta osin kehittämään.

Sovelluksen lopullisen käyttöliittymän valmistuttua suoritettiin lopullinen käytettävyystestaus. Tähän vaiheeseen osallistui viisi henkilöä. Thomas Landauerin [37] esittämän kaavan mukaan viiden testihenkilön käyttäminen on vaatimiinsa resursseihin

sekä tuloksiin nähden riittävän tehokas tapa kartoittaa järjestelmän toiminnallisuus.

Käytettävyystestaukseen osallistuneet henkilöt saivat myöhemmin täytettäväkseen lomakkeen, jonka avulla pyrittiin arvioimaan sovelluksen käytettävyyttä. Kysely jakautui kahdentyyppisiin kysymyksiin. Ensimmäisessä osassa käytettiin viiden pisteen Likert-asteikkoa arvioimaan kokemuksia erittäin kielteisistä erittäin myönteisiin. Toisessa osassa käyttäjiä pyydettiin itse määrittämään järjestelmän hyviä ja huonoja puolia. Kyselyn avulla pyrittiin löytämään järjestelmän mahdolliset ongelmakohdat ja tarvittaessa kehittämään niitä.

Kyselyn yhteydessä kartoitettiin myös käyttäjien suhtautuminen rajattuun ja avoimeen HOPSiin. Tästä saatua palautetta on kuitenkin vaikea arvioida testiryhmän koon ja koulutustaustan takia. Kyselyyn vastanneilla opiskelijoilla on ollut mahdollisuus tutustua ainoastaan rajatun HOPSin lähestymistapaan, joten riittävä vastakkainasettelu avoimeen HOPSiin puuttui.

## 6.2 Käyttöönotto

Järjestelmä otettiin varsinaisesti julkiseen käyttöön vaiheittain maaliskuun 2005 aikana. Se asennettiin Lappeenrannan teknillisen yliopiston tietotekniikan osaston WWW-ohjelmistopalvelimelle. Tässä vaiheessa varsinainen tuotantopalvelin ja kehityspalvelin erotettiin toisistaan. Kaikki järjestelmään mahdollisesti myöhemmin suoritettavat muutokset toteutetaan ensin kehityspalvelimella, josta ne testaamisen jälkeen siirretään tuotantokäyttöön. Loppuvuoden 2004 ja alkuvuoden 2005 ajan järjestelmä oli ollut testausryhmän käytettävissä.

Ensimmäinen osasto, jonka opiskelijoille tarjottiin mahdollisuus järjestelmän käyttöön oli tietotekniikan osasto. Järjestelmään oli toteutusvaiheen aikana mallinnettu LTY:n tietotekniikan opintosuunnan tutkintorakenteet vuodesta 1999 alkaen. Samalla osaston opintoneuvonnan sivuille lisättiin linkki järjestelmään.

Kevään ja alkukesän 2005 aikana järjestelmän tietokantaan lisättiin myös konetekniikan osaston tutkintorakenteet vuodesta 2001 alkaen. Lisääminen toteutettiin osaston omien resurssien toimesta. Samassa yhteydessä suoritettiin järjestelmän hallintaan luodun käyttöliittymän lopullinen testaus. Saadun palautteen perusteella sovellusta kyettiin yksinkertaistamaan ylläpitäjän kannalta.

Kaikki järjestelmään tai sen tietokantaan myöhemmin suoritettavat muutokset tulee toteuttaa kehityspalvelimella ennen niiden varsinaista käyttöönottoa. Ne testataan kehitysympäristössä ja ne voidaan siirtää tuotantokäyttöön hallitusti vasta sen jäl-

keen, kun riittävän huolellinen testaus on suoritettu. Huolimattomasti ja kiireessä tehdyt muutokset voivat aiheuttaa suuren riskin käytössä olevaan järjestelmään.

## 6.3 Käytettävyyspalautteen arviointi

Järjestelmän toimivuutta ja sen hyödyllisyyttä pyrittiin kartoittamaan käyttäjille suoritetun kyselyn avulla. Sen avulla pyrittiin selvittämään sekä käytettävyyteen liittyviä seikkoja että järjestelmän mahdollista tarpeellisuutta ja toimivuutta. Kyselyyn vastanneiden taustaa ja suhtautumista HOPSiin pyrittiin myös kartoittamaan, jotta mahdollisesti voitaisiin huomata, mikäli erityylisten käyttäjäryhmien antama palaute poikkeaa toisistaan. Käyttäjille lähetetty palautelomake on esitetty liitteessä 4.

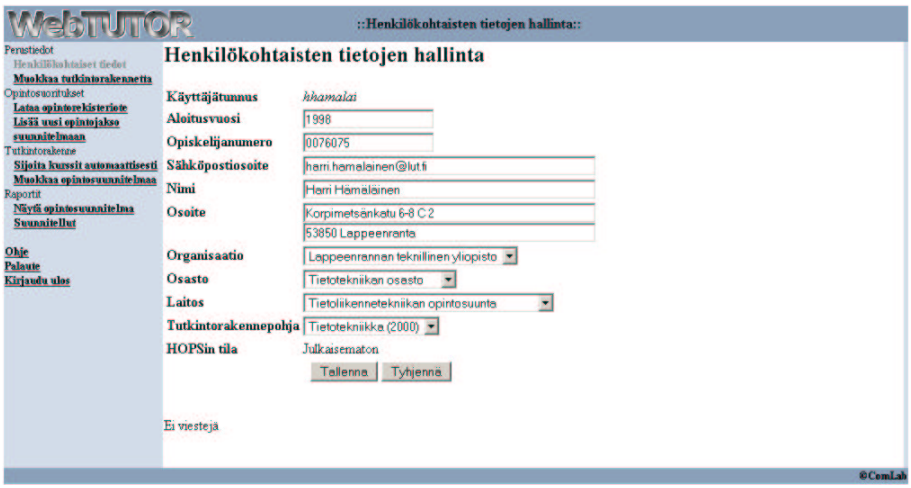
Käyttäjiltä saatu palaute oli erittäin positiivista ja järjestelmä koettiin hyödylliseksi työkaluksi opintosuunnitelmaa laadittaessa ja ylläpidettäessä. Pääosa testaajista oli suorittanut opintoja yli 100 opintoviikon verran, joten jokainen oli ainakin jossain opintojensa vaiheessa joutunut miettimään opintorakennettaan. Puolet kyselyyn vastanneista ei kuitenkaan aiemmin ollut laatinut itselleen opintosuunnitelmaa.

Järjestelmän avulla toteutetun opintosuunnitelman luomiseen meni käyttäjillä aikaa keskimäärin hieman alle tunti. Niiltä käyttäjiltä, jotka eivät olleet aikaisemmin laatineet HOPSia, keskimääräinen aika oli hieman yli tunnin. Tämä on kuitenkin huomattavasti vähemmän, kuin se aika, jonka jo aiemmin suunnitelman laatineet käyttäjät arvioivat kuluneen HOPSinsa laatimiseen perinteisellä tapaa. Käyttäjät mielsivät opintojen automaattisen sijoittelun tutkintorakenteeseen tehokkaaksi apuvälineeksi. Hyödylliseksi koettiin myös se, että suorittamatta tai aikatauluttamatta olevat opintojaksot esitettiin opintoryhmittäin, jolloin ne oli helppo lisätä osaksi omaa suunnitelmaa. Myös osittain näiden toiminnallisuuksien aikaansaama ajansäästö koettiin positiivisena asiana.

Käyttäjien kannalta hankalimmaksi koettiin opintosuoritusten siirtäminen järjestelmään. Tämä aiheuttaa opiskelijoille ylimääräistä päänvaivaa, sillä he joutuvat opintosuorituksia järjestelmään tallentaessaan hakemaan ne itse ulkopuolisesta järjestelmästä. Myös tukea useammalle yhtäaikaiselle HOPSille toivottiin.

## 6.4 Case: Oman opintosuunnitelman luominen

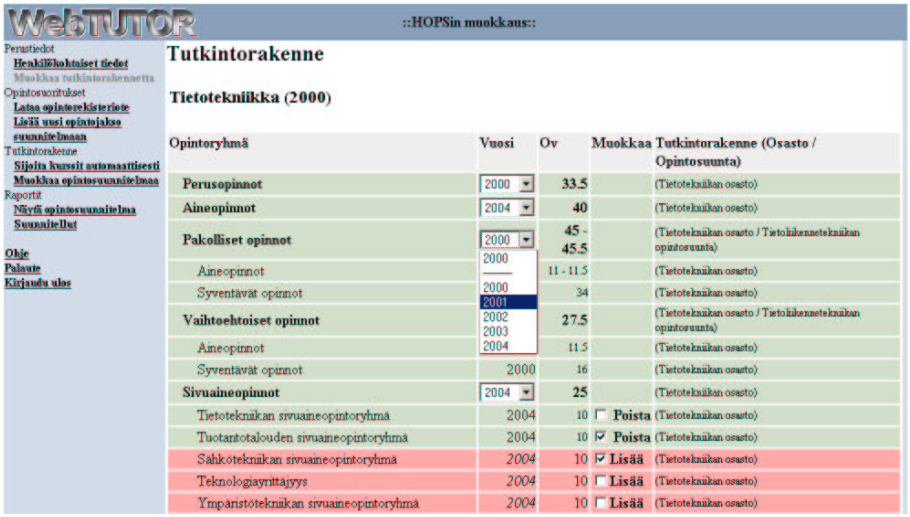
Opintosuunnitelman luominen aloitetaan täyttämällä henkilökohtaiset tiedot järjestelmän tietokantaan. Näiden tietojen lisäksi valitaan osasto, opintosuunta sekä HOPSin pohjana käytettävä tutkintorakennepohja halutulta vuodelta. Mikäli tässä vaiheessa ei vielä ole tietoa tulevasta opintosuunnasta, voidaan valita se, jota myöhemmin uskotaan opiskeltavan. Tarvittaessa tietoja voi vaivattomasti muuttaa myöhemmin, mikäli tiedot muuttuvat. Tietojen syöttämisen jälkeen ne tallennetaan sellaisenaan tietokantaan. Henkilökohtaisten tietojen syöttölomake on esitetty kuvassa 6.1.




Kuva 6.1: Henkilökohtaisten tietojen hallinta.

Kun tutkintorakennepohja on valittu ja muut tiedot syötetty, voidaan oman HOPSin rakennetta muokata tarkemmin. Seuraavassa vaiheessa voidaan tutkintorakenteen opintokokonaisuuksia vaihtaa vastaavien, eri vuosilta olevien, kokonaisuuksien välillä. Samalla valitaan ne vapaaehtoiset ryhmät, kuten sivuaineopinnot, jotka sisällytetään omaan HOPSiin. Tarvittaessa myös näitä tietoja voidaan myöhemmin muokata omaa tutkintoa paremmin palvelevaksi kokonaisuudeksi. Tutkintorakenteen muokkausnäkymä on osittain esitetty seuraavalla sivulla kuvassa 6.2. Tutkintoon sisällytetyt ja sisällyttämättömät vapaaehtoiset opintoryhmät on havainnollistettu eri värein.

Kun opintosuunnitelman runko on luotu, voidaan suoritetut opintojaksot siirtää järjestelmään. Opinnot voidaan siirtää joko lähettämällä Oodista tallennettu ote palvelimelle tai syöttämällä yksittäisen kurssin suoritustiedot manuaalisesti. Tämän jälkeen järjestelmään siirretyt opintojaksot voidaan pyrkiä automaattisesti sijoitta-




Kuva 6.2: Tutkintorakenteen muokkausnäkymä.

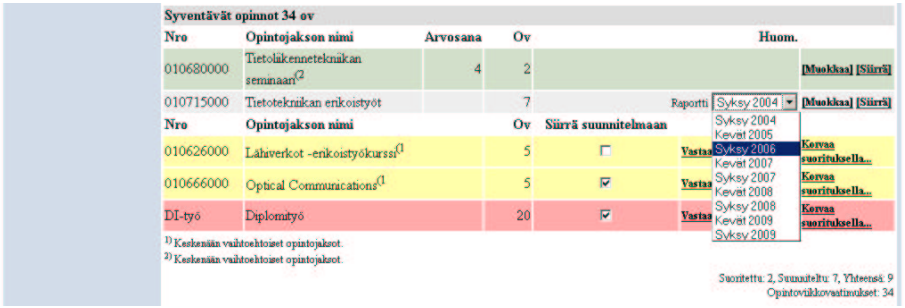
maan aiemmin luotuun tutkintorakenteeseen. Kun järjestelmä on sijoittanut opinnot rakenteeseen, ilmoittaa se käyttäjälle tehtävästä suoriutumisesta sekä sijoitettujen kurssien määrästä.

Seuraava vaihe HOPSin laatimisessa on siirtää itse loput sijoittamattomista kursseista oikeisiin opintoryhmiin. Jokaisen sijoittamattoman opintojakson kohdalla on pudotusvalikko, jossa on kuvattu HOPSin rakenne kokonaisuudessa opintoryhmineen. Kullekin sijoitettavalle opintojaksolle valitaan ryhmä, jonne se siirretään. Näkymä opintojen sijoittamisesta ja pudotusvalikossa havainnollistetusta tutkintorakenteesta on esitetty kuvassa 6.3.


Kuva 6.3: Opintojaksojen sijoittaminen HOPSiin.

Kun halutut opintojaksot on siirretty oikeisiin ryhmiin, lisätään mahdolliset korvaavuudet tai vastaavuudet vaadittujen ja jo suoritettujen opintojaksojen välille. Tämän jälkeen voidaan lisätä opintosuunnitelmaan jo suoritettujen opintojen lisäksi myöhemmin suoritettaviksi aiotut opinnot. Opintojaksot havainnollistetaan käyttäjälle eri värein. Kuvassa 6.4 on esitetty yksittäisen opintoryhmän sisältämät opintojaksot. Kyseiseen ryhmään on sisällytetty kaksi opintojaksoa, jotka ovat esitetty vihreällä ja harmaalla värillä. Vihreällä merkitty on suoritettu ja harmaalla pohjalla esitetty opintojakso on suorittamatta, joten sen suoritus voidaan aikatauluttaa.

Opintosuunnitelmasta puuttuvat tutkintorakenteen mukaiset opintojaksot on esitetty kuvassa 6.4 punaisella ja keltaisella värillä. Punainen väri merkitsee, että opintojakso kuuluu pakollisena suorituksena kyseiseen opintoryhmään. Keltaisella värillä vastaavasti esitetään ne opintojaksot, jotka ovat vapaavalintaisia kyseisessä ryhmässä. Ne voidaan siirtää osaksi omaa HOPSia lisäämällä rasti ruutuun kyseisten opintojaksojen kohdalle. Ne voidaan myös haluttaessa korvata sisällöltään niitä vastaavalla kurssilla. Mikäli puuttuva kurssi halutaan hakea korvattavaksi toisella kurssilla, jonka sisältö ei suoranaisesti vastaa kyseistä opintojaksoa, voidaan myös näin menetellä. Kyseisessä kuvassa HOPSiin kuulumattomat vapaaehtoiset kurssit ovat lisäksi keskenään vaihtoehtoisia, jolloin ainoastaan toinen niistä tulee sisällyttää osaksi HOPSia. Myöhemmin tietoja voidaan muokata, lisätä vastaavuuksia tai siirtää opintojaksoja toisiin ryhmiin.




Kuva 6.4: Opintoryhmän hallinta.

Kun HOPS on muokattu lopulliseen muotoonsa, voidaan se lähettää arvioitavaksi ja tulostaa. Opiskelija saa tulosteena rakenteeltaan vaatimusten mukaisen dokumentaation, jota voidaan käyttää opintojen aikana HOPSina sekä myöhemmin lopullisena valmistumissuunnitelmana. Järjestelmällä toteutettu lopullinen opintosuunnitelma on esitetty liitteessä 5.

# Luku 7

# Johtopäätökset

Opintosuunnitelman ja henkilökohtaisen hyväksytettävän tutkintorakenteen viilaaminen sääntöjen mukaiseksi vaatii usein aikaa ja vaivaa. Tässä työssä toteutetun järjestelmän avulla on tutkintorakennepohjaisen opintosuunnitelman laatiminen kyetty tekemään aikaisempaa helpommaksi ja tehokkaammaksi. Samalla opiskelijoiden opintosuunnitelmille tarjotaan keskitetty tietovarasto, jossa se säilyy tallennettuna. Samalla mahdollistetaan kaikille järjestelmän avulla luoduille suunnitelmille yhtenäinen ulkoasu, jonka avulla tarkastamista voidaan myös nopeuttaa, mikäli tarkastaminen halutaan suorittaa perinteisesti paperilta. Järjestelmän tietokantaan tallennettua suunnitelmaa opiskelija voi päivittää pidemmänkin tauon jälkeen vaivatta.

Järjestelmä häivyttää opintosuunnitelman ja opintojen loppuvaiheessa hyväksytettävän tutkintorakenteen välisen eron. Opiskelijalla on mahdollisuus päivittää opintojensa alussa luotua HOPSia opintojensa edetessä. Osa opintojen alussa suunnitelluista kursseista on voinut jossain vaiheessa poistua suunnitelmasta ja uusia lisätty niiden tilalle. Nämä ovat myöhemmin hyväksytetty osaksi omaa suunnitelmaa ja tutkintoa.

Järjestelmä antaa opiskelijalle mahdollisuuden opintojen suunnittelemiseen koko opintojen ajaksi. Mitä tarkemmin opiskelija tietää mitä haluaa, sitä tarkemmin hän voi suunnitelmansa laatia. Huolellisen suunnittelun avulla opiskelija voi joka tapauksessa luoda opinnoistaan suunnitelman, joka muodostaa mielekkään kokonaisuuden. Kokonaisuutta palvelevat opinnot suoritetaan oikeassa järjestyksessä ja kuormittavuutta pyritään hallitsemaan järkevällä aikatauluttamisella. Opiskelija osaa suunnata resurssinsa oikeisiin kohteisiin ja pitkällä aikavälillä säästää sekä omaa aikaansa että yliopiston resursseja. Kaikki tämä toiminta tähtää lopulliseen laadukkaaseen

tutkintoon.

Toteutettu järjestelmä ja sen mahdollistamat toiminnot voivat toimia mallina opintojensuunnittelun nykyaikaistamiseen tähtäävien järjestelmien kehitystyössä. Käyttäjiltä kerätyn palautten perusteella voidaan todeta, että erilaisten teknisten ratkaisujen avulla voidaan mekaanisen työn määrää vähentää oleellisesti ja samalla parantaa sovelluksen käytettävyyttä. Selkeä esimerkki tällaisesta ratkaisusta on suoritettujen opintojaksojen automatisoitu sijoittaminen rakenteeseen omaan tutkintorakennepohjaan. Työssä keskityttiin lisäksi tarkastelemaan WWW-pohjaisille järjestelmille asetettavia tietovaatimuksia sekä pyrittiin esittämään ratkaisut niiden estämiseen.

Opiskelijoilta saadun palautteen perusteella toteutetuntyyppiselle palvelulle on tarvetta. HOPSin vaivaton luominen ja ylläpitäminen madaltaa kynnystä nähdä opintosuunnitelma itselleen hyödyllisenä suunnitelmana, ei niinkään pakollisena dokumenttinä. Opintosuunnitelma on usein ollut dokumentti, joka on luotu opintojen alkuvaiheessa ja seuraavan kerran sitä on päivitetty juuri ennen valmistumista. Tämäntyylisestä suunnittelusta on tarve päästä eroon ja saada HOPS luonnolliseksi osaa opintojen etenemistä opintorekisterin rinnalle.

Järjestelmä ei luonnollisesti poista opiskelijan omaa vastuuta HOPSin luomisesta ja ylläpitämisestä. Mitä enemmän opiskelijalla on vapauksia valita, sitä suurempi vastuu omien opintojen suunnittelusta opiskelijalla on. Myös vastuu suunnitelman noudattamisesta ja opintojen suorittamisesta säilyy edelleen opiskelijalla.

Suoritetun käytettävyystestauksen perusteella sovelluksessa tulisi kehittää opintosuoritusten automaattista siirtämistä järjestelmään. Koska testiryhmän jäsenille tietotekniikka oli tuttua, tämän ongelman on helppo kuvitella korostuvan entisestään, mikäli järjestelmää käyttävät myös ei-teknologiaorientoituneet opiskelijat.

Lisäksi muiden toimintojen ja palvelujen integroimista vastaavanlaisiin järjestelmiin kannattaa kartoittaa. Haasteita sovellusten väliselle integraatiolle aiheuttaa pääsääntöisesti se, etteivät järjestelmät tarjoa rajapintoja tai niiden toteuttaminen vaatii huomattavan määrän resursseja. Yhteinen rajapinta Lappeenrannan teknillisellä yliopistolla epävirallisesti käytössä olevan *Lukkarimaatti*-sovelluksen tai muiden vastaavien järjestelmien kanssa saattaisi hyvinkin toiminnoiltaan tukea molempien järjestelmien käyttäjiä. Oman opintosuunnitelman pohjalta automaattisesti luodun periodikohtaisen lukujärjestyksen voisi hyvin sisällyttää tarjottavaksi palveluksi.

Jotta sähköinen tarkastaminen käytännössä olisi mahdollista, vaatii se myös tarkastavan tahon vahvan sitoutumisen järjestelmän käyttöön. Kuten tavallista, uuden järjestelmän käyttöönotto vaatii organisaatiolta resursseja, kun henkilökuntaa jou-

dutaan kouluttamaan järjestelmän käyttöön. Muutosvastarinnasta selvittäessä sähköinen tarkastaminen tarjoaa mahdollisuuden ohjaamisen tehostamiseen ja ajansäästöön niin opiskelijoiden kuin ohjaajienkin kannalta.

# Lähteet


[1] Valtioneuvosto. Pääministeri Matti Vanhasen hallituksen ohjelma. Koulutus, tiede ja kulttuuri. Valtioneuvoston kanslia, Helsinki 2003. 55 s.

[2] Kokko, Ossi. eHOPO ja muita HOPS-kokemuksia suomen kielen laitoksella. Valtakunnallinen HOPS-seminaari, Kuopion yliopisto 13.4.2005. (Seminaariesitelmä)

[3] Laitinen, Antti. Joustava opettajuus muuttuvissa konteksteissa. Jyväskylä: Korpi-Jyvä Oy, 1994.

[4] Ansela, Martti; Haapaniemi, Tommi & Pirttimäki, Säde. Yliopisto-opiskelijan hops. Ohjaajan opas. Kuopio: Kevama, 2005. 87 s. ISBN 951-781-479-8.

[5] Auer, Antti. eOPAS ja eHOPS suunnittelun ja ohjauksen välineinä. TieVie asiantuntijakoulutus, Jyväskylän yliopisto 12.-13.8.2004. (Seminaariesitelmä ja luentokalvot, 18 s.)

[6] Opetusministeriö. Bolognan prosessi [verkkodokumentti]. [viitattu 5.5.2005]. Saatavissa: `http://www.minedu.fi/opm/koulutus/yliopistokoulutus/bolognaprosessi.html`.

[7] Pietikäinen, Jussi. Tiedonsiirtorajapintojen vertailu. Esimerkkinä Desmond. Oodi-konsortio, 29.3.2004. 3 s.

[8] Saarinen, Vesa. Re: Oodin rajapinnoista [yksityinen sähköpostiviesti]. Vastaanottajat: Harri Hämäläinen, Markku Pääkkö (cc), Jukka Kurvi (cc). Lähetetty 8.3.2005 klo 16:37 (GMT +0200).

[9] Kalermo, Salla & al. Kuikka-projekti. Vaatimusmäärittely. Jyväskylän yliopisto, 1.4.2004. 18 s.

[10] Hyvönen Ilmari & Natunen, Kari. eHOPS - Projektisuunnitelma. Oodi-konsortio, 19.4.2005. 17 s.

[11] Hyvönen, Ilmari; Natunen, Kari & Tanninen, Jani. eHOPS - Vaatimusluettelo. Oodi-konsortio, 13.4.2005. 10 s.

[12] Garcia-Molina, Hector; Ullman, Jeffrey D. & Widom, Jennifer. Database Systems. The Complete Book. New Jersey: Prentice Hall, 2002. 1119 s. ISBN 0-13-098043-9.

[13] Nielsen, Jacob. Designing Web Usability: The Practice of Simplicity. Indianapolis: New Riders Publishing, 2000. 419 s. ISBN 1-56205-810-X.

[14] Hix, Deborah & H. Rex Hartson. Developing user interfaces: Ensuring usability through product and process. New York: John Wiley and Sons, 1993. 412 s. ISBN 0-471-57813-4, ISSN 0736-6906.

[15] Song, Ruihua & al. Learning important models for web page blocks based on layout and content analysis. ACM SIGKDD Explorations Newsletter, 2004 Vol. 6: 2. S. 14–23.

[16] Miller, R. B. Response time in man-computer conversational transactions. Proceedings of AFIPS Fall Joint Computer Conference. Montvale, New Jersey, USA, December 1968. Volume 33. S. 267–277.

[17] Freier, Alan O.; Karlton, Philip & Kocher, Paul C. The SSL protocol version 3.0. [verkkodokumentti] Transport Layer Security Working Group, 1996. Päivitetty 18.11.1996 [viitattu 12.2.2005]. Saatavissa: `http://wp.netscape.com/eng/ssl3/`.

[18] Verisign Inc. Internationalized Domain Names [verkkojulkaisu]. 2005 [viitattu 2.3.2005]. Saatavissa: `http://www.verisign.com/products-services/naming-and-directory-services/naming-services/internationalized-domain-names/index.html`.

[19] Johanson, Eric The state of homograph attacks, Rev 1.1 [verkkodokumentti]. Julkaistu 2005, päivitetty 2.11.2005 [viitattu 2.11.2005]. Saatavissa: `http://www.shmoo.com/idn/homograph.txt`.

[20] Kolsek, Mitja. Session Fixation Vulnerability in Web-based Applications. [verkkodokumentti] Acros - The Digital Security Research Lab, 2002. December

2002 [viitattu 3.2.2005]. Saatavissa PDF-tiedostona: `http://www.acros.si/papers/session_fixation.pdf`.

[21] Scott, David & Sharp, Richard. Abstracting Application-level Web Security. International World Wide Web Conference: Proceedings of the 11th international conference on World Wide Web. Honolulu, Hawaii, USA 7-11 May, 2002. ACM, 2002. S. 396–407. ISBN 1-58113-449-5.

[22] Shiflett, Chris. Foiling Cross-Site Attacks. PHP Architect [verkkolehti]. 2003, vol. 11: 10 [viitattu 3.4.2005]. Lehti ilmestyy myös painettuna. Saatavissa PDF-tiedostona: `http://www.phparch.com/issuedata/articles/article_66.pdf`.

[23] Huang, Yao-Wen & al. Securing Web Application Code by Static Analysis and Runtime Protection. Proceedings of the 13th international conference on World Wide Web. New York, NY, USA 17-22 May, 2004. ACM, 2004. S. 40–52. ISBN 1-58113-844-X.

[24] Haavisto, Juhani. Re: Desmondista taas [yksityinen sähköpostiviesti]. Vastaanottajat: Anni Rytkönen, Harri Hämäläinen, Jussi Ylikoski (cc). Lähetetty 28.6.2004 klo 18.17 (GMT +0300).

[25] Välimäki, Kari. Fw: Oodista ja eHOPSista Vastaanottajat: Jukka Kurvi, Harri Hämäläinen (cc), Tuija Huovila (cc). Lähetetty 7.3.2005 klo 13.52 (GMT +0200).

[26] Salo, Jussi & Välimäki, Kari. Tietojärjestelmäavusteinen opinto-oppaan teko. Sotka, opinto-opastietokanta ja opinto-oppaan julkaisujärjestelmä. Lappeenrannan teknillinen korkeakoulu, 2002. 37 s.

[27] Turkia, Johanna. Re: Sotkasta [yksityinen sähköpostiviesti]. Vastaanottaja: Harri Hämäläinen. Lähetetty 23.6.2004 klo 9.08 (GMT +0300).

[28] RFC 2865. Remote Authentication Dial In User Service (RADIUS). The Internet Engineering Task Force (IETF), 2000. 76 s.

[29] Bos, Bert & al. Cascading Style Sheets, level 2 revision 1. [verkkodokumentti] W3C Working Draft, 2005. Päivitetty 13.6.2005 [viitattu 17.9.2005]. Saatavissa: `http://www.w3.org/TR/CSS21/` (etusivu).

[30] Welling, Luke & Thomson, Laura. PHP and MySQL Web Development. First edition. Indianapolis: Sams Publishing, 2001. 896 s. ISBN 0-672-31784-2.

[31] W3C Working Group. Extensible Markup Language (XML) [verkkodokumentti]. Julkaistu 2003, päivitetty 29.10.2005 [viitattu 1.11.2005]. Saatavissa: `http://www.w3.org/XML/`.

[32] Olsen, Ole Kasper. Security Aspects Of a PHP/MySQL Based Login System for Web Sites. [verkkodokumentti] Opera Community. Julkaistu 23.2.2004 [viitattu 10.11.2004]. Saatavissa PDF-tiedostona: `http://my.opera.com/community/articles/php/securitysystem/secaspects.pdf`.

[33] Jacobs, Ian. URIs, Addressability ja the use of HTTP GET and POST. [verkkodokumentti] World Wide Web Consortium, 2003. Päivitetty 19.9.2004 [viitattu 2.3.2005]. Saatavissa: `http://www.w3.org/2001/tag/doc/whenToUseGet-20030919`.

[34] Jouravlev, Michael. Redirect After Post. [verkkodokumentti] TheServerSide.com, 2004. August 2004 [viitattu 20.12.2004]. Saatavissa: `http://www.theserverside.com/articles/article.tss?l=RedirectAfterPost`.

[35] Salminen, Harri. Kurssitiedon metatietomäärityksen XML-skeema. Tieteen tietotekniikan keskus, CSC, 2004 [viitattu 7.10.2004]. Saatavissa XSD-tiedostona: `http://meta.tv.funet.fi/schemaDocs/kurssitieto_xsd`.

[36] Booth, David & al. Web Services Architecture. [verkkodokumentti] W3C Working Group, 2004. Päivitetty 11.2.2004 [viitattu 11.11.2004]. Saatavissa: `http://www.w3.org/TR/ws-arch/`.

[37] Landauer, Thomas K. & Nielsen, Jakob. A mathematical model of the finding of usability problems. Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems. Amsterdam, The Netherlands 24-29 April, 1993. S. 206–213. ISBN 0-89791-575-5.