TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Elli Ylinen

TEOREETTISEN SÄHKÖTEKNIIKAN SIMULOINTIOHJELMAN KEHITTÄMINEN

Työn ohjaaja Jari Mikkolainen

Työn teettäjä Tampereen ammattikorkeakoulu

Työn valvoja Lauri Hietalahti

Tampere 2006

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikka
Ohjelmistotekniikka
Ylinen, Elli Teoreettisen sähkötekniikan simulointiohjelman kehittäminen
Tutkintotyö 33 sivua + 6 liitesivua
Työn ohjaaja Jari Mikkolainen
Työn teettäjä Tampereen ammattikorkeakoulu, valvojana Lauri Hietalahti
toukokuu 2006
Hakusanat sähkötekniikka, simulointi, Internetohjelmointi

Alkusanat

Keväällä 2005 Teoreettisen sähkötekniikan simulointiohjelman ideoija lehtori Lauri Hietalahti tarjosi minulle mahdollisuutta osallistua tämän ohjelman kehittämiseen. Suoritin insinöörin tutkintoon kuuluvan harjoittelun tätä ohjelmaa kehittämällä, joten tutkintotyön tekeminen tästä aiheesta oli itsestään selvä valinta.

Ohjelma toimii tällä hetkellä osoitteessa http://hph.tpu.fi/etesti, ja sitä on tarkoitus testata opetuskäytössä syksyllä 2006.

Kiitän lehtori Lauri Hietalahtea mahdollisuudesta työskennellä tämän ohjelman keittämiseksi. Kiitän myös Ilkka Hakkaria ja Onni Ylistä PHP-kielen opastuksesta.

Tampereella toukokuussa 2006

Elli Ylinen

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikka
Ohjelmistotekniikka
Ylinen, Elli Teoreettisen sähkötekniikan simulointiohjelman kehittäminen
Tutkintotyö 33 sivua + 6 liitesivua
Työn ohjaaja Jari Mikkolainen
Työn teettäjä Tampereen ammattikorkeakoulu, valvojana Lauri Hietalahti
toukokuu 2006
Hakusanat sähkötekniikka, simulointi, Internetohjelmointi

Tiivistelmä

Opetusmateriaalin siirtäminen sähköiseen muotoon lisääntyy jatkuvasti ja Internetin käyttöä opetuksessa hyödynnetään paljon. Sähkötekniikan alalla sen käyttö rajoittuu yleensä joko teorian jakeluun tai kuvaajia piirtävään ohjelmaan.

Tämän tutkintotyön tarkoitus on ollut muokata teoreettisen sähkötekniikan simulointiohjelman kokeellisesta versiosta kansainväliseen opetuskäyttöön soveltuva versio. Tavoitteena on ollut Internetissä toimiva sähkötekniikan oppimiseen käytettävä dynaaminen ohjelma, joka sisältää sähkötekniikan ilmiöiden teoriaa ja simulointityökaluja yhdistettynä yhdeksi kokonaisuudeksi.

Tässä työssä esitellään ohjelman muokkaamisen vaiheita kielen valinnan, teoriaosuuden lisäämisen ja käyttäjien hallinnan osalta. Ohjelmaan toteutetaan myös uusi simulointityökalu osoitinpiirroksia varten.

Ohjelma toimii vaatimusten mukaisesti ja sitä tullaan testaamaan opetuskäytössä syksyllä 2006.

Jatkokehityssuunnitelmiin kuuluu ohjelmasta aloitettava kansainvälinen projekti, jonka tarkoitus on kääntää ohjelma uusille kielille ja kehittää ohjelmaa edelleen.

TAMPERE POLYTECHNIC
Computer Systems Engineering
Software Engineering
Ylinen, Elli Development of the theoretical electric engineering simulation program
Engineering Thesis 33 pages, 6 appendices
Thesis Supervisor Jari Mikkolainen
Commissioning Tampere polytechnic Supervisor: Lauri Hietalahti
May 2006
Keywords electric engineering, simulation, Internet programming

Abstract

Internet is used in educational purpose in many schools. In electric engineering Internet usually contains only theoretical part. There are also simulation tools for electric engineering in the Internet. The aim of this study was to develop the demo version of theoretical electric engineering simulation program to international education tool. The aim was to have a program in the Internet, which includes theory of electric engineering phenomena and possibility to simulate them. This study is about modifying the program so that it can carry out several users at the same time and the language can be easily changed and translated. This study introduces also a new simulation tool and addition of a new theoretical part.

Program works as demanded and will be tested in educational use. The continuing development plan includes intention about international co-operation whit this project in the future.

SISÄLLYSLUETTELO

Termit ja lyhenteet

TESTI	Teoreettisen sähkötekniikan simulointiohjelmasta muodostettu lyhenne.
HTML	HTML eli Hyper Text Markup Language määrää web-sivujen sisällön rakenteen. HTML:ää käytetään tekstin ja kuvien esittämiseen halutulla tavalla Internet-sivuilla sekä linkityksiin muille Internet-sivuille./2/
CSS	CSS eli Cascading Style Sheets määrää tyyliasetukset eli kuinka sivujen sisältö esitetään. CSS kootaan omaksi tiedostokseen, sijoitetaan HTML:n <HEAD> -elementtien sisään tai sisällytetään HTML-elementtiin. /2/
PHP	PHP eli Hypertext Preprosessor on avoimen lähdekoodin scriptikieli, joka on kehitetty erityisesti dynaamisten Internet-sivujen tekemiseen./1/
MySQL	MySQL on avoimen lähdekoodin hallintajärjestelmä relaatiotietokannalle. MySQL-tietokantaa käsitellään sql-kielellä ja ohjelmalogiikka voidaan toteuttaa useilla kielillä./3/
SQL	SQL eli Structured Query Language on tietokantojen hallintaan kehitetty kieli./2/
JavaScript	JavaScript on vapaasti käytettävissä oleva suosittu skriptikieli, joka voidaan kirjoittaa suoraan HTML-koodin sisälle. /2/
Istunto	Tässä työssä istunnolla tarkoitetaan PHP:n sessioiden hallintatekniikalla tehtyä käyttäjän yksilöllistä istuntoa./1/
Gnuplot	Gnuplot on komentotulkkipohjainen grafiikan piirtoon tarkoitettu ohjelma./4/

1 Johdanto

Internet on nykyään yleisesti käytössä kouluissa ja kaikilla ammattikorkeakoulun opiskelijoilla on mahdollisuus käyttää sitä opiskeluunsa. Oppimateriaalin siirto sähköiseen muotoon on lisääntynyt jatkuvasti. Enää kaikkien kurssien oppimateriaalia ole saatavilla muuten kuin Internetistä tai koulun intranetistä. Suomen virtuaaliammattikorkeakoulu lähti liikkeelle virallisesti marraskuussa 2001, jolloin solmittiin yhteistyösopimus suomalaisten ammattikorkeakoulujen välillä. Yksi osa tämän organisaation strategiaa on hyödyntää nykyaikaista tieto- ja viestintätekniikkaa, jolloin opiskelija voi tietokoneen avulla hoitaa koko opiskelunsa missä tahansa maailmassa./5/ Koulut ovat jo useiden vuosien ajan siirtäneet oppimateriaalia verkkoon. Perinteisillä aloilla, kuten sähkötekniikassa, materiaali rajoittuu yleensä oppikirjamaiseen teoriaosuuteen. Internetistä löytyy kyllä myös simulointityökaluja, jotka piirtävät kuvaajan, mutta teoria täytyy etsiä toiselta sivustolta.

Teoreettisen sähkötekniikan simulointiohjelman idea lähti liikkeelle Tampereen ammattikorkeakoulun sähkötekniikan osaston lehtori Lauri Hietalahden ideasta opettaa sähkötekniikan ilmiöitä havainnollistaen kuvin ja kuvaajin. Oppilaille hankalat asiat olivat visualisoinnin kautta entistä helpommin opittavissa ja opetettavissa, koska teorian lisäksi sivuilta pääsisi helposti katsomaan ilmiön kuvaajaa ja osoitinkuvia. Innoittajana tähän kansainväliseksi suunniteltuun projektiin toimi ohjelmistojen verkko-opetusmateriaalia tekevä Codewitz-projekti, joka sai alkunsa Tampereen ammattikorkeakoulussa vuonna 2000 ja on sen jälkeen saanut yhteistyökumppaneita ympäri maailmaa.

Lehtori Hietalahti yhdessä vaihto-oppilas Lampros Chaidaksen kanssa suunnitteli TESTI-ohjelmasta Internetissä toimivan kokeiluversion, jonka Chaidas toteutti. Suunnittelun tärkeitä lähtökohtia olivat dynaamisuus ja havainnollisuus sekä se, että ohjelmaa voivat käyttää kaikki, jotka haluavat oppia sähkötekniikkaa. Ohjelman tulee olla vapaassa levityksessä, joten se päätettiin toteuttaa käyttäen ilmaisia työkaluja. Lampros Chaidaksen vaihto-oppilasvuoden päättyessä ohjelman jatkokehitys siirtyi tämän tutkintotyön aiheeksi.

Teoreettisen sähkötekniikan simulointiohjelmaa voidaan käyttää sähkötekniikan oppimateriaalina ja simulointityökalun kautta myös ilmiöiden havainnollistamiseen sekä jatkokehitystavoitteiden mukaan myös harjoitustehtävien kautta tapahtuvaan oppimiseen.

1.1 Työn lähtökohdat

Ohjelma toimi hyvin yhdellä käyttäjällä ja pohja oppisivuja varten oli luotu. Gnuplot-ohjelma oli linkitetty ohjelmaan ja MySQL-tietokanta oli luotu käyttäjien rekisteröimiseksi. Ohjelma oli tehty kaksikieliseksi, mutta suomenkielisen osuuden käyttäjälle tulostuvat tekstit olivat suurimmaksi osaksi englanniksi. Kielille oli tehty kansiot, joissa oli samat HTML- ja PHP-tiedostot. Ohjelman toimintaa tutkittiin edellisen ohjelmoijan ja työn ideojan kanssa ja ohjelman jatkokehitystavoitteita suunniteltiin. Edellinen ohjelmoija oli keskittynyt kehittämään käyttäjälle mahdollisuutta omien yhtälöiden perusteella piirrettäviin Gnuplot-ohjelman kuvaajiin. Tämän osuuden jatkokehitys siirrettiin kuitenkin

pitkälle tulevaisuuteen ja keskityttiin ohjelmaan tuleviin valmiisiin esimerkkitapauksiin. Päätettiin, että ohjelmasta muokataan versio, joka voidaan ottaa opetuskäyttöön. Työn tilaajan kanssa sovittiin myös, että kielenkääntäjältä voidaan vaatia HTML-osaamista, mutta ei PHP- tai SQL-taitoja. Tärkeitä lähtökohtia olivat myös ohjelman edelleen kehittäminen tämän tutkintotyön jälkeen ja käyttöönoton mahdollistaminen muissa maissa.

1.2 Aiheen rajaus

Tässä opinnäytetyössä käsitellään usean käyttäjän yhtäaikaisen käytön mahdollistaminen ja kielenvalinnan muokkaaminen. Lisäksi työssä käsitellään osoitinkuvien dynaaminen piirtäminen ja teoriakirjaston rakentaminen. Gnuplot-ohjelman toiminnasta on suppea yleiskuvaus ja Gnuplot-ohjelmalla piirrettävien kuvaajien yksilöinti käsitellään erillään käyttäjäryhmistä. Ohjelman tietoturvan tai käyttäjän syötteiden tarkistamisen ratkaisuja tässä työssä vain sivutaan, mutta niitä ei tarkemmin käsitellä. Lisäksi tässä työssä valotetaan joitakin ohjelman ulkonäköön tai tietosisällön lisäämiseen liittyviä muutoksia. Sähkötekniikan materiaalin ja laskukaavat toimittaa työn tilaaja, joten niiden sisältöön ei tässä työssä perehdytä.

2 Ohjelman kuvaus ja toteutettavat muutokset

TESTI-ohjelmasta oli tehty Internetissä toimiva opetusväline, jolla voitiin sähkötekniikan teorian opiskelun lisäksi piirtää kuvaajia sähkötekniikan ilmiöistä. TESTI-ohjelmaa oli alusta asti ohjelmoitu PHP:lla, koska se on dynaamisten Internet-sivujen tekemiseen erityisesti kehitetty skriptikieli/1/. PHP:n dokumentaatio on helposti saatavilla Internetistä, joten se oli itsestään selvä valinta myös ohjelman jatkokehitykseen. Myös muut lähtötilanteessa TESTI-ohjelmassa olleet tekniikat todettiin toimiviksi myös jatkokehitykseen.

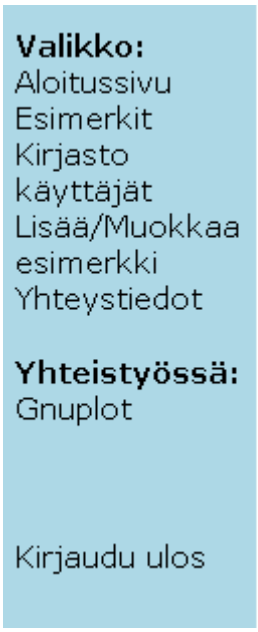
2.1 Yleistä

Lähtötilanteessa TESTI-ohjelmaan vaadittiin sisäänkirjautuminen, minkä jälkeen käyttäjä pääsi käyttämään ohjelmaa. Sisäänkirjautuminen haluttiin säilyttää TESTI-ohjelmassa, koska sen avulla käyttäjät voidaan tunnistaa. TESTI-ohjelman näkymä käyttäjälle oli koostettu kolmesta HTML-sivusta, eli ohjelman näkymä oli toteutettu HTML:n kehysten avulla. Vasemmalla on valikko, josta käyttäjä saattoi valita haluamansa toiminnon ja ylhäällä oli paneeli, josta käyttäjä voi valita haluamansa kielen. Nämä kaksi sivua olivat koko ajan esillä käyttäjälle. Kolmannessa ikkunassa oli muuttuva näkymä, jonka määrää käyttäjän valikosta valitsema toiminto. Tämä näkymä todettiin hyväksi ja se säilytettiin. TESTI-ohjelman koko näkymästä on kuva liitteenä 1.

TESTI-ohjelma vastaanottaa käyttäjien syötteet HTML-lomakkeiden kautta. Arvot välitetään PHP:n POST-metodilla, joka on yksi PHP:n tapa siirtää dataa HTML-lomakkeesta PHP-koodille/1/. Tätä tapaa käytettiin jo lähtötilanteessa ja se todettiin toimivaksi, joten sitä käytettiin kaikkien HTML-lomakkeiden arvojen välityksessä.

2.2 Valikko

Valikko (kuva 1) muutetaan käyttäjäkohtaiseksi, jotta käyttäjälle voidaan näyttää käyttöoikeuksiensa mukaiset vaihtoehdot. Kun käyttäjä kirjautuu ohjelmaan, tarkistetaan käyttäjän käyttöoikeudet ja valikko säilyy muuttumattomana käyttäjän näkymässä uloskirjautumiseen asti.



Kuva 1 Valikko

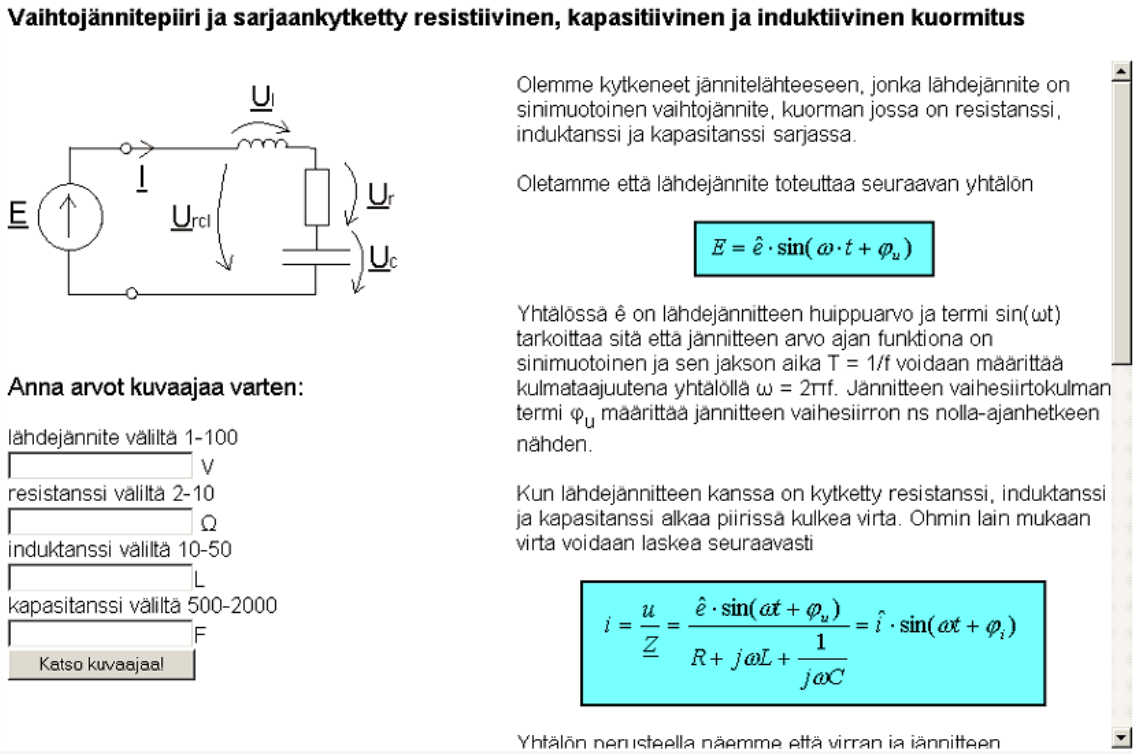
2.2.1 Aloitussivu

Aloitussivu-linkistä käyttäjä pääsee aina takaisin ensimmäiselle kirjautumisen jälkeen näytettävälle sivulle. Tämän linkin toimintaa ei ollut tarpeen muokata, koska käyttäjälle haluttiin säilyttää mahdollisuus palata helposti ohjelman esittelysivulle. Aloitussivun PHP- ja HTML-tiedostoja muokattiin mahdollistamaan kielivalinta ja käyttäjien kirjautuminen. Nämä muutokset käsitellään kielivalinnan yhteydessä luvussa kuusi.

2.2.2 Esimerkit

Lähtötilanteessa TESTI-ohjelman opetusmateriaali suunniteltu näytettäväksi käyttäjälle muuttuvan näkymän kehyksessä ja malli esimerkkipiirin näkymästä oli toteutettu. TESTI-ohjelmaan lisättiin kuvan 2 mukaisia esimerkkejä sähköteknisten piirien ilmiöistä, jotta tietosisältöä saatiin lisättyä. Esimerkit-linkistä käyttäjä pääsee tarkastelemaan valitsemiaan sähköteknisten piirien ilmiöitä. Esimerkit-linkin taakse lisättiin sisällysluettelo, johon lisätään linkki kaikille ohjelmaan tehdyille valmiille esimerkkipiireille, jotta käyttäjä voi valita, mitä haluaa tutkia ja simuloida. Käyttäjälle aukeaa näkymä, joka koostuu oppikirjamaisesta teoriaosuudesta, Gnuplot-ohjelman käyttöliittymästä ja piirikaaviosta (kuva 2). Näkymää uudelleen suunniteltaessa haluttiin, että esimerkkipiirin piirikaavio ja Gnuplot-ohjelman käyttöliittymä ovat koko ajan

käyttäjän nähtävissä. Tästä syystä teoriaosuus sijoitettiin vierityspalkilla varustettuun inline-kehykseen. Piirikaavio ja Gnuplot-ohjelman käyttöliittymä saadaan näin pidettyä käyttäjän näkymässä, vaikka teoriaosuutta luetaan vierityspalkin avulla.



Kuva 2 Esimerkkipiirin näkymä, joka sisällytetään muuttuvan näkymän kehykseen

2.2.3 Kirjasto

Linkki toteutettavaan teoriakirjastoon lisätään valikkoon, koska haluttiin, että se on koko ajan käyttäjän näkymässä. Kirjasto sisältää sähkötekniikan suureiden määritelmiä ja sääntöjä, joita ei selitetä yksityiskohtaisesti kaikkien esimerkkien yhteydessä. Nämä haluttiin kerätä yhteen, jotta käyttäjä voisi käyttää ohjelmaa helposti myös määritelmien ja sääntöjen tarkistamiseen. Kirjasto-linkin taakse lisättiin sisällysluettelo helpottamaan halutun määritelmän tai säännön etsintää. Näihin tehdään linkitys myös esimerkkien teoriaosuuden avainsanoista nopeuttamaan ja helpottamaan käyttäjän tiedonhakua. Kirjaston toteutus löytyy luvusta 7.

2.2.4 Käyttäjien hallinta

Lähtötilanteessa käyttäjien hallinnan linkki oli näkyvillä kaikille käyttäjille. Tätä linkkiä muokataan niin, että sen kohdalla näkyy käyttäjän TESTI-ohjelman käyttöoikeuksien mukainen linkki. Pääkäyttäjä on ainoa, jonka tarvitsee päästä käsittelemään kaikkia käyttäjätietoja, joten muille käyttäjäryhmille tämä linkki on tarpeeton. Käyttäjäryhmät käsitellään luvussa 3. Kuvan 1 valikko on pääkäyttäjän ja ohjelman ylläpitäjän valikko.

2.2.5 Yhteystiedot

Yhteystiedot-linkin takaa käyttäjä löytää ylläpitäjän ja käyttäjätietojen muokkaukseen oikeutettujen henkilöiden tiedot. Yhteystiedot-sivulle päivitettiin ajankohtaiset tiedot, mutta muilta osin se todettiin toimivaksi ja tarpeelliseksi, eikä se vaatinut enempää muokkausta.

2.2.6 Gnuplot-ohjelman linkki

TESTI-ohjelma hyödyntää kuvaajien piirrossa Gnuplot-ohjelmaa, joten kiinnostuneet käyttäjät haluttiin ohjata linkin kautta Gnuplot-ohjelman kotisivuille, jossa he voivat tutustua Gnuplot-ohjelman mahdollisuuksiin. Gnuplot-ohjelmasta kerrotaan tarkemmin luvussa 4.

2.2.7 Uloskirjautuminen

TESTI-ohjelmaan täytyy tehdä myös uloskirjautuminen, jotta tiedetään, koska käyttäjästä tallennetut tiedot voidaan tuhota. Uloskirjautuminen lopettaa käyttäjän istunnon ja tyhjentää käyttäjän tiedot sisällään pitävän taulukon. Myös selaimen sulkeminen toteuttaa saman asian. Käyttäjän istunnot käsitellään luvussa 3.

2.3 Kielen vaihtaminen

Käyttäjä voi valita TESTI-ohjelman käyttökielen suomen kielen ja englannin kielen välillä. TESTI-ohjelmaa oli alusta asti tehty kaksikieliseksi, koska sen haluttiin olevan mahdollisimman monen käyttäjän käytettävissä. Ohjelma vaati kielen valinnan ennen sisään kirjautumista, jotta osattiin näyttää kirjautumislomake oikealla kielellä. Käyttäjä voi halutessaan vaihtaa kielen yläkehyksen kieliä kuvaavista lipuista. Liput ovat kuvassa 3 ja koko näkymän kuvassa, joka on liitteenä 1. Mahdollisuus kielen vaihtamiseen kirjautumisen jälkeen haluttiin säilyttää, jotta käyttäjällä on mahdollisuus vertailla sähkötekniikan sanastoa eri kielillä ilman uutta kirjautumista. Kielen vaihtaminen muokattiin TESTI-ohjelmaan niin, että PHP-koodi on sijoitettu omaan kansioonsa ja sillä helpotettiin kielenkääntäjän työtä. Kielivalinnan muokkaamisesta kerrotaan luvussa 6.

Kuva 3 Liput yläkehyksessä

2.4 Ulkonäköasetukset

Ohjelman CSS-kielellä tehtyihin ulkonäköasetuksiin tehtiin vain pieniä muutoksia ja ohjelmaan tulevissa lisäyksissä noudatettiin samankaltaista edellisen ohjelmoijan määrittelemää ulkonäköä. Lisäyksiä ja muutoksia tarvittiin

esimerkkipiirejä lisättäessä, koska kaikille tarvittaville tyyleille ei ollut määrittelyjä. Muutoksia tehtiin vain tekstin ulkonäköön ja CSS-tyylimääritysten sijaintiin. CSS-kieli siirrettiin pois HTML-tiedostoista ja koottiin kolmeksi kehyksittäiseksi CSS-tiedostoksi. Tämä tehtiin helpottamaan jatkokehityksen yhteyteen suunniteltua ulkonäkön muokkausta, joka on tarkoitus teettää Internet-sivujen tyyleihin perehtyneellä opiskelijalla.

3 Käyttäjien hallinta

Lähtötilanteessa ohjelmassa oli sisäänkirjautuminen ja käyttäjät yksilöitiin tietokantaan, mutta kuvaajien piirto oli mahdollista vain yhdelle käyttäjälle kerrallaan. Jotta ohjelmaa voitaisiin käyttää opetukseen, sen täytyy pystyä käsittelemään useita käyttäjiä yhtä aikaa. Koska TESTI-ohjelman tavoitteeksi asetettiin sen hyödyntäminen opetuskäyttössä, nähtiin tarpeelliseksi eritellä oppilaskäyttäjä ja opettajakäyttäjä käyttöoikeuksien kannalta. Suunnittelun edetessä nähtiin tarpeelliseksi myös pääkäyttäjä, joka olisi myös ohjelman ylläpitäjä.

3.1 Käyttäjät-taulu

Tietokanta sisälsi yhden taulun, jossa käyttäjien tiedot säilytettiin.

Käyttäjät-taulun rakenne ennen muokkausta oli seuraavanlainen:

id etunimi sukunimi käyttäjätunnus salasana sähköpostiosoite

Käyttäjäryhmien luontiin tarvittiin vielä yksi sarake lisää käyttäjät-tauluun kertomaan, onko kyseinen käyttäjä oppilas, opettaja vai pääkäyttäjä, koska TESTI-ohjelman piti pystyä tunnistamaan käyttäjän tiedoista tämän käyttäjäryhmä. Tunnistus tarvittiin, jotta TESTI-ohjelma osaa antaa käyttäjälle oikean määrän oikeuksia. Tauluun lisättiin sarake, jossa on jokaiselle käyttäjäryhmälle oma kokonaislukuarvo. Kokonaislukuarvoa käytettiin käyttäjäryhmien erottamiseen, koska sitä oli helppo hyödyntää PHP:lla toteutetuissa käyttäjäryhmien tarkistuksissa. TESTI-ohjelmaa suoritettaessa voitiin näitä hyödyntäen tarkistaa käyttäjän oikeudet tietokantaan tallennetusta käyttäjäryhmätiedosta.

Käyttäjät-taulu sisälsi muokkauksen jälkeen tiedon käyttäjäryhmästä:

id etunimi sukunimi käyttäjätunnus salasana sähköpostiosoite ryhmä

3.2 PHP:n istunnot

TESTI-ohjelman dynaamisuuden aikaansaamiseksi käytettiin PHP:n istuntojen hallintatekniikkaa. Istunnot valittiin, koska se oli yksinkertaisin tapa muokata lähtötilanteen TESTI-ohjelma käyttäjäkohtaiseksi. Istunnoissa on myös hyvä tietoturva ja taulukko käyttäjän yksilöllisille muuttujille, joita tarvitaan käyttäjän tietojen varastoimiseen. Istunnolla on yksilöllinen tunnus, josta käyttäjä voidaan tunnistaa. Istunto alkaa käyttäjän kirjautuessa ohjelmaan ja päättyy vasta käyttäjän

kirjautuessa ulos ohjelmasta tai sulkiessa selaimen. Taulukon tiedot säilyvät, vaikka käyttäjä siirtyy ohjelmassa sivulta toiselle, ja tämä mahdollistaa dynaamisten sivujen tekemisen. /1/

3.3 Istunnon muuttujat

Sisäänkirjautuminen ohjelmaan alkaa käyttäjän valitessa haluamansa kielen ohjelman aloitussivulta. Kielivalinta on yksilöllistä tietoa käyttäjästä ja siksi luotiin istunnon taulukkoon muuttuja, joka sisälsi valitun kielen. Tämä oli välttämätöntä, koska tietoa valitusta kielestä tarvitaan, kun TESTI-ohjelman näkymää vaihdetaan. Kielivalinnan toteutus löytyy luvusta neljä. Kun kieli on valittu, ohjelma vaihtoi näytettäväksi sivuksi kirjautumissivun, jonka HTML-lomakkeen kautta käyttäjän tunnus ja salasana välitetään tunnistettavaksi. Tunnistus suoritetaan, jotta suunniteltu kaikkien käyttäjien rekisteröinti voitiin toteuttaa. Käyttäjän kirjauduttua oikeilla tunnuksilla, lisätään käyttäjän käyttäjätunnus istunnon taulukon muuttujaan, jotta TESTI-ohjelmalla on tieto käyttäjän onnistuneesta kirjautumisesta. Tätä tietoa tarvittiin käyttäjän istunnon käynnistämiseksi ja oikean näkymän näyttämiseksi. Käyttöoikeuksien tarkistamiseksi piti istunnon taulukkoon lisätä muuttuja, jonka avulla käyttäjälle näytetään käyttöoikeuksiensa mukainen sisältö ohjelmasta. Käyttäjäryhmän kertova kokonaisluku sijoitetaan istunnon muuttujaan, jotta sitä voidaan hyödyntää näkymän valinnassa.

PHP-esimerkki:

```
<frame src="<?php echo $_SESSION['lang'];?>/left<?php
echo $_SESSION['usergroup'];?>.html" name="left">
```

Yllä olevassa esimerkissä on TESTI-ohjelman aloitussivun HTML- ja PHP-koodia. Esimerkin koodi asettaa TESTI-ohjelman käyttäjäkohtaisen valikon käyttäjän näkymään onnistuneen kirjautumisen jälkeen. Esimerkissä muuttujat ovat istunnon taulukon muuttujia, jotka sisältävät käyttäjän valitseman kielen ja käyttäjälle annetun käyttäjäryhmän. Nämä sijoitettiin HTML-koodin keskelle, jolloin niiden sisältämät muuttujat tulostuivat HTML-koodin sisään. Tämä mahdollisti käyttäjäkohtaisen valikon näyttämisen käyttäjälle. Esimerkki tulostuu pelkäksi HTML-koodiksi seuraavasti.

HTML-esimerkki:

```
<frame src="finnish/left2.html" name="left">
```

Yllä olevassa esimerkissä käyttäjä on käyttäjäryhmässä kaksi, joka näkyy tulostettuna valikon HTML-tiedoston nimeen. Käyttöoikeuksien mukaisen valikon nimi muodostetaan näin myös muille käyttäjäryhmille. Käyttäjän on tässä tapauksessa valinnut suomen kielen, mikä sijoitetaan valikon kehyksen lähdetiedoston tiedostopolkuun. Kielivalinnasta kerrotaan tarkemmin luvussa kuusi.

3.4 Käyttäjien oikeudet

Käyttöoikeuksien myöntämisestä päätettiin, että jokaiselle käyttäjälle myönnetään vain välttämättömät oikeudet. Tällä haluttiin minimoida TESTI-ohjelman väärinkäytön riskejä.

3.4.1 Oppilaskäyttäjä

Oppilaskäyttäjälle annettiin oikeudet vain ohjelman avulla tapahtuvaan opiskeluun, koska oppilaskäyttäjän ei tarvitse nähdä tai muokata käyttäjätietoja. Oppilaskäyttäjä saa käyttäjätunnuksen ja salasanan, joilla pääsee kirjautumaan TESTI-ohjelmaan, selaamaan teoriaosuutta ja simuloimaan sähkötekniikan ilmiöitä.

3.4.2 Opettajakäyttäjä

Opettajakäyttäjälle annettiin kaikki samat oikeudet, kuin oppilaskäyttäjällä on. Opettajakäyttäjän oikeuksiin lisättiin oppilaskäyttäjien tietojen rajoitettu selaus. Tiedoista ovat saatavilla vain yksilöivä tunnus (id), nimi ja sähköpostiosoite, koska muut tietokannassa olevat tiedot eivät ole välttämättömiä opettajakäyttäjälle. Opettajakäyttäjille annettiin myös oikeudet lisätä ja poistaa uusia tunnuksia oppilailleen, jotta he voivat hyödyntää ohjelmaa opetuksessaan. Opettajakäyttäjälle ei myönnetty oikeuksia valita lisättävän käyttäjän käyttäjäryhmää, vaan TESTI-ohjelma asettaa automaattisesti opettajakäyttäjän lisäämille tunnuksille oppilaskäyttäjän oikeudet. Opettajakäyttäjän oikeuksien myöntäminen haluttiin säilyttää pääkäyttäjällä, jotta kaikki käyttäjät, joilla on oikeuksia TESTI-ohjelman tunnusten lisäämiseen tai poistamiseen, ovat pääkäyttäjän tiedossa. Pääkäyttäjä voi tällöin myös ennen oikeuksien myöntämistä varmistaa, onko uudella käyttäjällä todellinen tarve muiden käyttäjien lisäämiseen.

3.4.3 Pääkäyttäjä

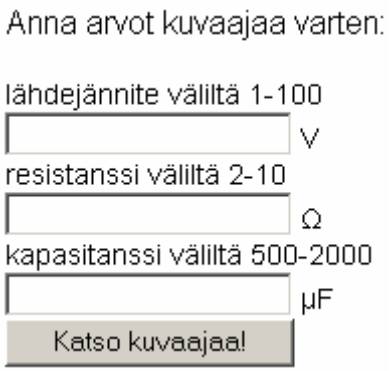
Pääkäyttäjälle annettiin ohjelman toiminnan turvaamisen kannalta tarvittavat oikeudet, jotta hän voisi ylläpitää TESTI-ohjelmaa. Pääkäyttäjä luo kaikki opettajakäyttäjät ja tarvittaessa myös oppilaskäyttäjiä. Pääkäyttäjä pääsee näkemään kaikki tiedot käyttäjistä tietokannasta, koska pääkäyttäjän velvollisuuksiin kuuluu myös unohtuneiden käyttäjätunnusten ja salasanojen korjaus. Pääkäyttäjän velvollisuuksiin päätettiin lisätä myös mahdollisen uuden kielen lisääminen TESTI-ohjelmaan, koska pääkäyttäjällä on oikeudet nähdä ja muokata TESTI-ohjelman lähdekoodia ylläpidon vaatimalla tavalla.

4 Gnuplot-ohjelma

Gnuplot-ohjelma on ilmainen komentotulkkipohjainen grafiikan piirtoon tarkoitettu ohjelma. Se on suunniteltu erityisesti visualisoimaan tieteellistä dataa. Gnuplot-ohjelmalla voidaan esimerkiksi piirtää kaksi- ja kolmiulotteisia kuvaajia annetuista funktioista sekä mittaustuloksista sovitettuja funktioita. Gnuplot-ohjelmalle voidaan antaa kuvaajien piirtoon tarvittavat komennot yksitellen ohjelman käyttöliittymässä tai kaikki kerralla komentotiedostona. Tässä tiedostossa voidaan piirrettävän funktion lisäksi määritellä palautettavan kuvan koko, koordinaatiston asteikot, kuvaajien värit ja useita muita ominaisuuksia. /4/

4.1 Gnuplot-ohjelman käyttö TESTI-ohjelmasta

Lähtötilanteessa TESTI-ohjelmassa oli Gnuplot-ohjelmalle käyttöliittymä, joka oli sijoitettu HTML-taulukon soluun. Solun näkymä vaihdettiin Gnuplot-ohjelman tuottamaan kuvaajaan, kun käyttäjä oli syöttänyt numeeriset arvot kuvaajan piirtämiseksi ja painanut HTML-lomakkeen painiketta. TESTI-ohjelma kirjoitti Gnuplot-ohjelmalle lähetettävän tekstitiedoston, joka sisälsi Gnuplot-ohjelman grafiikan piirtoon vaatimat komennot. Gnuplot-ohjelma käynnistettiin PHP-koodista ja samalla lähetettiin komennot sisältävä tekstitiedosto argumentiksi. Gnuplot-ohjelma piirsi komentojen mukaisen kuvaajan, joka palautettiin kuvana käyttäjän selaimelle HTML-lomakkeen paikalle. Kuvassa 5 nähdään Gnuplot-ohjelman tekemä kuva.



Kuva 4 Muokattu HTML-lomake arvojen syöttämiseen.

TESTI-ohjelmaan tarvittiin uusia Gnuplot-ohjelmalle komentotiedostoja kirjoittavia PHP-tiedostoja kattamaan lisättyjen esimerkkipiirien vaatimia kuvaajia. Tämä toteutettiin muokkaamalla valmiista Gnuplot-ohjelman komentotiedostosta PHP-tiedosto, jossa Gnuplot-ohjelman komennot kirjoitetaan uudeksi Gnuplot-ohjelman komentotiedostoksi ja halutut sähkötekniikan suureet korvattiin muuttujilla, joihin sijoitettiin käyttäjän syöttämät arvot. Tämä toteutettiin lähtötilanteen PHP-tiedostoa mallina käyttäen, koska ratkaisu oli todettu toimivaksi. Kaikille lisätyille esimerkkipiireille toteutettiin myös tarvittavat arvot vastaanottava HTML-lomake arvorajoineen (kuva 4). Nämä muutokset toteuttivat käyttäjälle suunnitellun mahdollisuuden simuloida kaikkia TESTI-ohjelmassa olevia esimerkkipiirejä.

4.2 Käyttöliittymän muokkaus

Lähtötilanteessa Gnuplot-ohjelman tuottaman kuvaajan ja arvojen syöttöön tehdyn HTML-lomakkeen vaihtamista HTML-taulukon solun näkymään ei ollut huomioitu ollenkaan. Gnuplot-ohjelman piirtämän kuvaajan yläpuolelle haluttiin HTML-taulukon solun näkymän vaihtava HTML-painike, jotta käyttäjä voisi vaihtaa käyttöliittymän näkymän kuvaajasta takaisin arvoja vastaanottavaan HTML-lomakkeeseen. Painikeen HTML-koodi lisättiin PHP-tiedostoon, joka kirjoittaa kuvaajan piirtämiseen vaadittavan PHP-koodin. Tämä PHP-tiedosto sisällytettiin arvojen syöttöön käytetyn HTML-lomakkeen paikalle jo lähtötilanteessa, joten lisätty painike saatiin sille suunniteltuun paikkaan Gnuplot-ohjelman tuottaman kuvan yläpuolelle ilman suuria muutoksia. Painiketta painettaessa käyttäjän näkymään vaihdetaan takaisin HTML-lomake, jolla arvot syötetään. Tämä toteutettiin niin, että painikkeen painaminen lataa uudestaan saman HTML-taulukon soluun sijoitetun PHP-tiedoston. PHP-koodi tutkii, ovatko

HTML-lomakkeen arvoja välittävät POST-metodin muuttujat olemassa. Näitä muuttujia ei lähetetä edelleen painikkeen tulostavalle PHP-tiedostolle, joten muuttujat eivät ole olemassa, kun tätä PHP-tiedostoa suoritetaan. Tulokseksi saadaan, että kun muuttujia ei ole olemassa, HTML-taulukon solussa on kuvaaja. Tällöin PHP-tiedosto vaihtaa solun sisällöksi HTML-lomakeen arvojen syöttämiseksi. Jos POST-metodin muuttujat ovat olemassa, suoritetaan Gnuplot-ohjelmalle komentotiedoston kirjoittava PHP-tiedosto.

PHP-esimerkki:

```
if(!isset($_POST['x']) && !isset($_POST['z']))
{
include("../$lang/accrcircuit/insertvalues.html");
}
else
{
    include('accrprosessor.php');
    echo '<img src="lowdiagram.php">';
}
```

Yllä oleva esimerkki on Gnuplot-ohjelman käyttöliittymän näkymän valitseva PHP-koodi. Painikkeen HTML-koodi on sisällytettävässä PHP-tiedostossa (Liite 2). Tämä toteutus oli yksinkertainen ja toimiva, joten sitä käytettiin TESTI-ohjelmassa.

5 Kuvaajien yksilöinti

Kuvaajien yksilöinnin apuna käytetään käyttäjän istunnon taulukkoa, jonka muuttujiin tallennetaan käyttäjän HTML-lomakkeessa antamat arvot. Näiden muuttujien avulla Gnuplot-ohjelman komentotiedostoon voitiin kirjoittaa käyttäjäkohtaiset arvot.

5.1 Käyttäjäkohtaisten arvojen syöttö

Käyttäjän syöttämille arvoille annettiin tietyt yleiset rajat, jotta kuvaajien havainnollisuus ja järkevyys säilyvät. Arvot myös tarkistettiin ennen Gnuplot-komentotiedoston luomista epäonnistuneiden kuvaajien välttämiseksi (Liite 5).

Käyttäjän painaessa HTML-lomakkeen painiketta, välittyvät HTML-lomakkeen tiedot PHP-koodin käsiteltäviksi. Arvot välitetään TESTI-ohjelmassa käytetyllä POST-metodilla, jonka taulukon muuttujista käyttäjän antamat arvot sijoitetaan tarkastusten jälkeen istunnon käyttäjäkohtaisiin muuttujiin (Liite 5). Näiden muuttujien avulla voitiin luoda käyttäjäkohtainen komentotiedosto.

5.2 Käyttäjäkohtainen komentotiedosto

Arvot sijoitettiin istunnon muuttujiin ja TESTI-ohjelmassa suoritettiin Gnuplot-ohjelman komentotiedoston kirjoitus sekä välitys Gnuplot-ohjelmalle, jotta kuvaaja voitiin piirtää ja näyttää käyttäjälle. PHP-koodi avaa käyttäjän komentotiedoston ja kirjoittaa sinne rivi kerrallaan Gnuplotia ohjaavia komentoja. Käyttäjän antamat arvot sijoitetaan ohjelmalogiikan avulla komentoihin. Ero lähtötilanteeseen verrattuna on käyttäjän istunnon muuttujan käyttö, POST-

metodin muuttujan sijaan. Tämä muutettiin, jotta käyttöliittymän näkymän vaihtava painike voitiin lisätä yksinkertaisesti ja käyttäjän syöttämiä arvoja voitaisiin käyttää myös osoitinpiirrosten tekemiseen. Osoitinkuvat käsitellään luvussa 7.

PHP-esimerkki:

```
$_SESSION['gnuc']=0,000005;
fwrite($file, "C(Td)="); fwrite($file,
$_SESSION['gnuc']); fwrite($file, " \r\n");
```

Tuloksena on Gnuplot-komentotiedostossa:

```
C(Td)=0,00005
```

Esimerkkikoodissa kirjoitetaan yksi rivi Gnuplot-komentotiedostoon. Käyttäjä on antanut kapasitanssille arvon, joka on tallennettu käyttäjän istunnon taulukon muuttujaan. Muuttujaa käytettiin PHP-koodissa mahdollistamaan Gnuplot-ohjelman komentojen muunneltavuutta. Kaikkia käyttäjän syöttämiä arvoja käytettiin vastaavalla tavalla Gnuplot-ohjelman komentotiedoston kirjoittamisessa. Yksi Gnuplot-ohjelman komentotiedoston kirjoittava PHP-tiedosto liitteenä 2 ja komentotiedosto liitteenä 3.

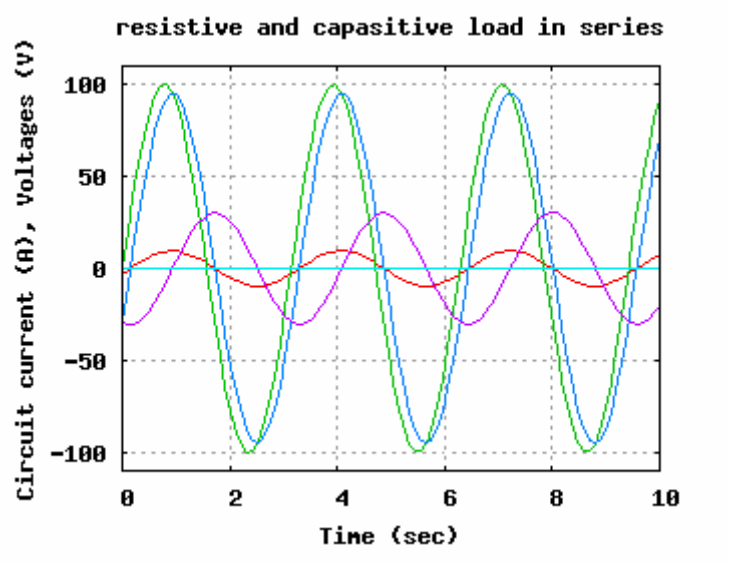
Gnuplot-ohjelman komentotiedostot täytyi myös nimetä käyttäjäkohtaisesti. Lähtötilanteessa TESTI-ohjelman komentotiedostot kirjoittava PHP-koodi avasi aina gnuplot.txt tiedoston, jonne komennot kirjoitettiin. Tämä täytyi muuttaa, koska samaan komentotiedostoon ei voi kirjoittaa usea käyttäjä yhtä aikaa eri arvoilla. Komentotiedon virheellisyys aiheuttaisi väärän tuloksen tai ei tulosta ollenkaan. Käyttäjätiedostojen nimeäminen ratkaistiin käyttäjän istunnon muuttujan avulla. Komentotiedoston nimessä säilytettiin teksti ”gnuplot.txt” ja eteen lisättiin käyttäjän istunnon tunnus.

PHP-esimerkki:

```
$id=$_SESSION[‘id’]."gnuplot.txt";


exec("/usr/bin/gnuplot $id");
```

Yllä olevassa esimerkissä muuttujaan `$id` sijoitetaan käyttäjäkohtaisen komentotiedoston nimi, joka koostuu käyttäjän istunnon tunnuksesta ja vakio-osasta ”gnuplot.txt”. Esimerkistä näkyy myös Gnuplot-ohjelman käynnistys PHP-koodista, ja yhdistetty komentotiedoston nimi annetaan argumentiksi Gnuplot-ohjelmalle, joka palauttaa ohjelmalle käyttäjän komentotiedostosta piirretyn kuvaajan. Komentotiedosto kirjoitetaan näiden rivien välissä (Liite 2).



Kuva 5. Gnuplot-ohjelman piirtämiä kuvaajia.

5.3 Kuvaajien määrittelyt

Gnuplot-ohjelman komentotiedostossa määritellään kuvaajien värit ja koordinaatiston asteikot. Lähtötilanteessa kuvaajille oli määritelty värit ja asteikoille oli määritelty vakiokoko. Asteikot haluttiin muuttaa riippuvaisiksi käyttäjän syötteistä. Kuvaajien havainnollisuuden säilyttämiseksi tämä oli välttämätöntä, koska käyttäjälle oli haluttu antaa mahdollisuus myös kuvaajien lähdejännitteen arvon määrittelemiseen.

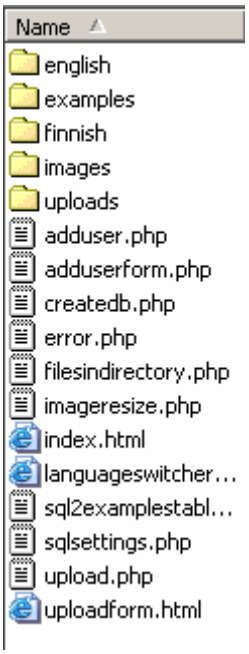
Lähdejännitteen arvoa käytettiin suhteellisen koon laskemisessa, koska se saa käyttäjän antamista arvoista suurimmat arvot kuvaajissa. Työn tilaaja halusi myös, että lähdejännitteen huiput olisivat selvästi näkyvissä, joten lähdejännitteen muuttujaa käytettiin kiinteän arvon sijaan komentotiedoston kirjoittavassa PHP-koodissa ja lisäksi muuttujalle lisättiin kerroin. Kertoimen vaikutus on nähtävissä kuvassa 5, jossa y-akselin asteikko on lähdejännite kerrottuna luvulla 1,1. Kertoimen ansiosta lähdejännitteen huippu on näkyvissä riippumatta lähdejännitteen suuruudesta ja asteikko on aina sopiva. Yksi PHP-tiedosto ja sen kirjoittama komentotiedosto ovat liitteinä 2 ja 3.

Gnuplot-ohjelman komentotiedostossa määritellään myös värit, joilla kuvaajat piirretään. TESTI-ohjelmaan harkittiin kuvaajien värin valinnan mahdollisuutta, mutta siitä luovuttiin. Todettiin, että havainnollisuus säilyy parhaiten kun värit määritellään vakioiksi. Tietyn värinen kuvaaja tarkoittaa TESTI-ohjelman jokaisessa Gnuplot-ohjelman piirtämässä kuvassa samaa suuretta. Tämä auttaa käyttäjää näkemään helposti kuvaajien suureet.

6 Kielivalinta

Koska ohjelman ympärille oli tarkoitus rakentaa kansainvälinen projekti, ohjelman täytyi toimia usealla kielellä. Lähtötilanteessa ohjelma oli kaksikielinen ja sisälsi kaksi erillistä kansiota nimettyinä ”finnish” ja ”english”. Kansioissa oli samat tiedostot nimettyinä yhdenmukaisesti, erona vain sivulle tulostuva kieli. Kielikansiosta toiseen pääsi yläkehyksen lippujen linkeistä. Ongelma tässä ratkaisussa oli, että osa PHP-koodia sijaitsi molemmissa kansioissa, joten

muutokset koodiin oli silloin tehtävä näiltä osin kahteen kertaan. Loput PHP-koodista sijaitsivat hakemistorakenteen juuritasolla ja esimerkkipiirejä varten luodussa kansiossa (kuva 6). Näissä ongelmana oli PHP-koodin sisällä oleva HTML-koodi, joka lähtötilanteessa oli mahdollista tulostaa käyttäjän näkymään vain toisella kielellä. Lisäksi uuden kielen lisäämiseen TESTI-ohjelmaan lähtötilanteessa tarvittiin henkilö, joka ymmärsi olla kääntämättä tai muuten muokkaamatta PHP-koodia tai SQL-lauseita.

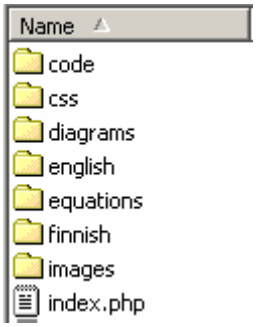


Kuva 6 Tiedostorakenne lähtötilanteessa.

6.1 Tiedostorakenteen muokkaus

Ongelman ratkaiseminen aloitettiin luomalla Code-kansio. Kansio sijaitsee samalla hakemistotasolla kielikansioiden kanssa ja sisältää PHP-koodin, joka on yhteistä molemmille kielille. PHP-kooditiedostot siirrettiin Code-kansioon, HTML-tiedostot jätettiin kielikansioihin ja hakemistopolut muutettiin suhteellisiksi. Code-kansion ansiosta muutokset PHP-koodiin saatiin tehtyä yhdessä tiedostossa, eikä muutoksia tarvinnut tehdä molempiin kielikansioihin. Code-kansioon tehtiin uusi kansio, johon esimerkkipiirien PHP-tiedostot siirrettiin. Tämä toteutettiin, jotta esimerkkipiirien lisääntyessä niiden PHP-tiedostot olisivat entistä helpommin löydettävissä. PHP-tiedostojen määrä kasvoi TESTI-ohjelmaa kehitettäessä niin paljon, että nähtiin tarpeelliseksi selkiyttää Code-kansion sisältämien PHP-tiedostojen tiedostorakennetta. Tämä nopeutti halutun PHP-tiedoston etsimistä ja nopeutti siten myös työskentelyä.

TESTI-ohjelmassa tarvittavat kuvat jaettiin omiin kansioihinsa (kuva 7). Jakaminen tehtiin kuvien käyttötarkoituksen perusteella. Esimerkkipiirien piirikaavioiden ja sähkötekniikan kaavojen kuvat siirrettiin omiin kansioihinsa. TESTI-ohjelman yleiset kuvat, kuten liput ja TESTI-logo, jätettiin omaan kansioonsa. Tällä ajateltiin kuvien määrän kasvua tulevaisuudessa ja siksi kuvat jaettiin käyttötarkoituksiensa mukaan eri kansioihin. Kansiot pidettiin tiedostorakenteen juuritasolla, jotta kuvien tulostaminen näkymään ei vaatisi pitkiä tiedostopolkuja kielien kansioista eikä Code-kansiosta.



Kuva 7. Tiedostorakenne muokkauksen jälkeen.

Tiedostorakenteen muokkauksen jälkeen kielikansio voidaan antaa kielenkääntäjälle, jonka ei tarvitse osata jättää kääntämättä kuin HTML-koodin merkinnät.

6.2 Kielikohtainen tekstin tulostus

Kielikohtainen tulostus käyttäjän valitseman kielen mukaan toteutettiin seuraavaksi. Useat tiedostot kielikansioissa sisälsivät ennen tiedostorakenteen muokkausta sekä PHP-koodia että näkymään HTML-koodilla tulostettavaa kieltä. Koska PHP-koodi oli kokonaan poistettu kielikansioista, tuli tarve luoda muuttuja, jonka avulla Code-kansion PHP-tiedostot voivat hakea tulostettavan tekstin oikeasta kansiosta. Tämä ratkaistiin luomalla istunnon taulukkoon muuttuja, joka pitää sisällään käyttäjän valitseman kielen ja samalla kielikansion nimen. Muuttuja voitiin tulostaa PHP-tiedostossa tiedostopolkuun, jonka avulla PHP-tiedosto etsii oikean HTML-tiedoston näkymään tulostettavaksi. Tämän toteuttaminen oli mahdollista, koska eri kielikansioissa sijaitsevat saman asian, mutta eri kielellä kertovat HTML-sivut on nimetty täsmälleen samoin.

PHP-esimerkki:

```
$lang=$_SESSION['lang'];
include("../$lang/accircuit/insertvalues.html");
```

Yllä olevassa esimerkissä sisällytetään valitulla kielellä oleva arvojen syöttöön tehty HTML-lomake näytölle. Lomake haetaan siirtymällä ensin hakemistorakenteen juureen ja sieltä muuttujan `$lang` osoittamaan kielikansioon ja edelleen kyseessä olevan esimerkkipiirin kansioon. Sieltä löytyy HTML-tiedosto, joka yllä olevassa esimerkissä on `insertvalues.html`. TESTI-ohjelma sisällyttää PHP-koodin avulla HTML-sivuja toisen HTML-sivun sisälle yllä olevan PHP-esimerkin mukaisesti. HTML-tiedostoja täytyi jakaa useiksi HTML-tiedostoiksi, jotta jokainen kielikohtainen tulostus käyttäjän näkymään voitiin tehdä oikealla kielellä. HTML-tiedostojen jakaminen toteutettiin ja jokaisesta erikseen näkymään sisällytettävästä kielikohtaisesta tulostuksesta tehtiin oma HTML-tiedosto. Tätä ratkaisua käytettiin, koska sen avulla voitiin PHP-koodi pitää yksinkertaisena, eikä se vaatinut monimutkaista tekstitiedostojen käsittelyä. Myöskään kielen sana- tai rivimäärällä ei ollut merkitystä tässä ratkaisussa, mikä oli tärkeää kielen kääntämisen kannalta.

6.3 Kielen vaihto

Lähtötilanteessa TESTI-ohjelmassa oli avaussivu, jossa näkymän muodosti vain TESTI-logo ja liput, joista kieli valittiin. Kun kieli oli valittu, siirryttiin sisäänkirjautumisnäkymään, jossa käyttäjä syötti tunnuksensa HTML-lomakkeessa. Tunnuksen käsittely tapahtui PHP-tiedostossa, jonne HTML-lomakkeen tiedot välitettiin. Jos käyttäjän tunnukset olivat tietokannassa, eli käyttäjä oli rekisteröity TESTI-ohjelman käyttäjäksi, muodostettiin kolmesta kehyksestä koostettu näkymä TESTI-ohjelmasta. Lähtötilanteessa yläkehyksen lipuissa oli linkki kielikansioiden aloitussivulle. Linkit olivat Suomen ja Englannin lippujen kuvat, joiden ulkoasu sopi hyvin TESTI-ohjelmaan, eikä niitä haluttu muuttaa.

TESTI-ohjelman kielenvaihtoa muuttui, johtuen kielikohtaisen tulostuksen muutoksista. Kielenvaihto vaati kielimuuttujan sisällön muuttamisen, jotta TESTI-ohjelma osaisi näyttää näkymät oikealla kielellä. Edelleenkin haluttu opiskelukieli täytyi valita jo ennen TESTI-ohjelmaan kirjautumista, koska käyttäjälle näytettävä kirjautumisen HTML-lomake oli tehty myös kielikohtaiseksi. Käyttäjällä haluttiin säilyttää mahdollisuus vaihtaa kieli myös kirjautuneena, jotta hän voisi verrata esimerkiksi englanninkielistä oppikirjaa suomenkieliseen TESTI-ohjelman teoriaosuuteen.

TESTI-ohjelman yläkehyksessä säilytettiin linkit kielen vaihtamiseksi, Kielikansioiden aloitussivut oli poistettu tiedostorakennetta muokattaessa, joten tämä ei enää onnistunut. Ongelma ratkaistiin yhdistämällä kolme käyttäjälle ensimmäiseksi näytettävää näkymää ohjattavaksi yhdestä PHP-tiedostosta.

Yhdistetyssä aloitussivun PHP-tiedostossa sisällytetään näkymään myös sisäänkirjautumisen kielikohtainen HTML-tiedosto. Näkymä valitaan istunnon muuttujien ja PHP:n POST- ja GET-metodien taulukoiden välityksellä tulleen tiedon perusteella. Tämä toteutettiin tarkistamalla PHP-koodissa oikeassa järjestyksessä muuttujien olemassa olo. Tämä PHP-tiedosto on liitteenä 4.

Muuttujat tarkistetaan TESTI-ohjelman aloitussivun PHP-tiedostossa viidellä kohdalla.

1.

Ensin tarkistetaan, onko käyttäjä halunnut kirjautua ulos. Jos on, istunnon muuttujat tyhjennetään ja käyttäjälle näytetään näkymä ennen sisäänkirjautumista ja kielen valitsemista. Jos käyttäjä ei ole painanut ulos kirjautumisen linkkiä, ei tiedon välittävässä GET-metodin taulukon uloskirjautumisen muuttujaa ole olemassa, joten PHP-tiedoston suoritus siirtyy seuraavaan kohtaan tekemättä mitään.

2.

Toisessa kohdassa tarkistetaan, onko HTML-lomakkeesta lähetetty POST-metodilla käyttäjän syöttämiä sisäänkirjautumisen tunnuksia. Jos on, tunnukset tarkistetaan ja asetetaan istunnon muuttujat käyttäjänimelle, käyttäjäryhmälle ja käyttäjän istunnon tunnukselle, jonka jälkeen PHP-tiedoston suoritus siirtyy

seuraavaan kohtaan. Jos tunnuksia ei ole lähetetty, siirrytään seuraavaan kohtaan tekemättä mitään.

3.

Kolmannessa kohdassa tarkistetaan, onko kieli valittu. Jos on, kielivalinta sijoitetaan istunnon kielimuuttujaan. Kielimuuttuja välityksessä käytetään PHP:n GET-metodia. GET-metodin taulukkoon tallennetaan tieto selaimen osoiteriviltä ja sitä voidaan käyttää suoraan PHP-koodissa/1/. Kun käyttäjä painaa linkistä, kirjoitetaan selaimen osoiteriville, millä kielellä TESTI-ohjelmaa halutaan käyttää. GET-metodin avulla tieto otetaan osoiteriviltä TESTI-ohjelman käyttöön.

PHP-esimerkki

```
if(isset($_GET['lang']))
{
    $_SESSION['lang']=$_GET['lang'];
}
```

Esimerkissä tutkitaan, onko kieli haluttu vaihtaa. Jos kielenvaihdon linkkiä ei ole painettu, siirrytään PHP-tiedoston suorituksessa seuraavaan kohtaan tekemättä mitään. Jos kieli on valittu, se tallennetaan istunnon kielimuuttujaan ja siirrytään seuraavaan kohtaan.

4.

Neljännessä kohdassa tarkistetaan, onko käyttäjänimi tallennettu istunnon muuttujaan. Jos on, käyttäjä on valinnut kielen ja kirjautunut sisään, joten käyttäjälle näytetään ensimmäinen TESTI-ohjelman näkymä kirjautuneelle käyttäjälle ja tämän PHP-tiedoston suoritus päättyy. Tässä kohdassa käytetään istunnon kielimuuttujaa oikean näkymän valitsemiseen. Käyttäjän on täytynyt valita kieli ennen käyttäjänimen tallentamista istunnon muuttujaan, koska sisäänkirjautumisen HTML-lomaketta ei muuten näytetä. Jos käyttäjänimeä ei ole tallennettu istunnon muuttujaan, siirrytään PHP-tiedoston suorituksessa seuraavaan kohtaan tekemättä mitään.

5.

Viides kohta tutkii, onko käyttäjän istunnolle asetettu kielimuuttuja. Jos on, näytetään käyttäjälle sisäänkirjautumisen HTML-lomake käyttäjän valitsemalla kielellä. Tämä tehdään, koska edellisten tarkistusten perusteella tiedetään, että käyttäjä ei ole vielä kirjautunut TESTI-ohjelmaan, koska PHP-tiedoston suoritus olisi muuten loppunut tarkistusten neljänteen kohtaan. Jos kielimuuttujaa ei ole olemassa, siirrytään PHP-tiedoston viimeiseen kohtaan, jossa ei tehdä enää tarkistuksia.

Jos PHP-tiedostoa suoritettaessa päästään tarkistusten jälkeen olevaan kohtaan asti, tarkoittaa se, että käyttäjä ei ole valinnut kieltä, eikä ole kirjautunut TESTI-ohjelmaan. Tällöin käyttäjälle näytetään TESTI-ohjelman näkymä, josta kirjautuminen aloitetaan kielen valinnalla, edellä mainituista syistä. Uloskirjautuminen ohjaa myös käyttäjän tähän näkymään, koska se tyhjentää

käyttäjän istunnon taulukon muuttujat, jotka tässä PHP-tiedostossa tarkistetaan. PHP-tiedosto on kokonaisuudessaan liitteenä 4.

Kielivalinnan muuttaminen TESTI-ohjelmaan kirjautumisen jälkeen ohjaa käyttäjän takaisin tähän PHP-tiedostoon(Liite 4). Kielivalinnan linkki osoittaa tälle sivulle ja valinnan muutos välitetään GET-metodilla, kuten myös ennen TESTI-ohjelmaan kirjautumista. PHP-tiedosto suoritetaan edellä kuvatulla tavalla ja täten tulokseksi saadaan TESTI-ohjelman kirjautumisen jälkeen näytettävä näkymä uudella kielellä. Käyttäjän muihin istunnon muuttujiin ei tehdä muutoksia, joten käyttäjälle ainoa muutos on kielen vaihtuminen.

Tätä ratkaisua käytettiin, koska se mahdollistaa uuden kielen lisäämisen helposti. Kielivalinnan säilyttäminen HTML-linkissä tarkoitti, että GET-metodi oli yksinkertaisin tapa toteuttaa kielen vaihtaminen. Istunnon muuttujan käyttäminen kielimuuttujana oli hyvä ratkaisu, koska sen täytyi olla käytettävissä koko TESTI-ohjelmassa. Istunnon muuttujaa käyttämällä tietoa käyttäjän kielivalinnasta ei tarvitse erikseen välittää jokaiselle PHP-tiedostolle.

6.4 Kielen lisääminen

Uuden kielen lisääminen TESTI-ohjelmaan on pääkäyttäjän velvollisuus, koska kielen lisäämiseen tarvitaan oikeudet TESTI-ohjelman lähdetiedostojen muokkaamiseen. Kielen lisäämiseen tarvitaan ensimmäiseksi kielikansio, jonka HTML-tiedostojen kieli on käännetty halutulle kielelle. Tämän jälkeen pääkäyttäjä lisää kielelle tätä kuvaavan lipun kuvan TESTI-ohjelman yleisten kuvien kansioon, jotta kuvan tulostuksen tiedostopolkuja ei tarvitse muuttaa. Lipulle lisätään linkki TESTI-ohjelman näkymään, joka näytetään ennen kirjautumista, jotta käyttäjä näkee kaikki valittavina olevat kielet. Kielenvaihtamiseksi TESTI-ohjelman suorituksen aikana, lisätään lippu linkiksi myös yläkehyksen HTML-tiedostoon. Tämä onnistuu helposti kopioimalla jonkin jo ohjelmassa olevan kielen linkin.

HTML-esimerkki

```
<a href="../index.php?lang=finnish"
target=_top><img src="../images/finnishsmall.png"
align="right"></a>
```

Pääkäyttäjän on tehtävä vielä kielivalinnan mahdollistavat muutokset yllä olevan esimerkin mukaiseen linkkiin. Kielimuuttujan arvoksi on vaihdettava kyseisen kielen, ja samalla kielikansion, nimi. Tämä on tärkeää, koska TESTI-ohjelma käyttää kielimuuttujan arvoa sanatarkasti kielikohtaisen tiedon esittämiseen. Koska kuva on lisätty TESTI-ohjelman yleisten kuvien kansioon, ei tiedostopolkua tarvitse muokata, vaan ainoastaan lippukuvan nimi täytyy vaihtaa uuden lippukuvan nimen mukaiseksi.

Näiden toimenpiteiden jälkeen ohjelma toimii lisätyllä kielellä, koska kielikansioiden sisältämät tiedostopolut on kirjoitettu suhteellisiksi ja Code-kansion PHP-tiedostojen koodissa käytetään istunnon muuttujaa, aina kun tarvitaan kielikohtaisia HTML-tiedostoja.

TESTI-ohjelman kielivaihtoehtojen lisääminen saatiin näillä ratkaisuilla yksinkertaiseksi. Ratkaisuilla täytettiin myös kielenkääntämisen vaatimus ja

käyttäjän kielivalinnan vaatimukset. Kielivalinnan ratkaisuissa ajateltiin myös TESTI-ohjelman tulevaisuutta, jotta tämän työn jälkeen tehtävässä TESTI-ohjelman jatkokehityksessä nämä ratkaisut voitaisiin säilyttää.

7 Teoriakirjasto

Lähtötilanteessa TESTI-ohjelmassa ei ollut teoriakirjastoa, joten tämän luominen oli kokonaan uusi osa TESTI-ohjelmaan. TESTI-ohjelman haluttiin simuloinnin ja ilmiöitä kuvaavien esimerkkien lisäksi kattavan myös sähkötekniikan teoriaa. Tätä tarkoitusta varten luotiin kirjasto sähkötekniikan määritelmille. Teoriakirjaston selaus haluttiin mahdollistaa myös esimerkkien opiskelun yhteyteen, siten että käyttäjän ei tarvitse poistua näkymästä toiseen teoriaa tarkastellakseen.

7.1 Linkit kirjastoon

Kirjastolle tehtiin sisällysluettelo ja linkki sinne lisättiin valikkoon. Tämä tehtiin, jotta kirjasto olisi hyvin esillä ja käyttäjän helposti löydettävissä. Kirjasto piti ensimmäisessä vaiheessa sisällään linkkiluettelon teoriaosuudesta. Jotta kirjaston sisältöä olisi helppo selata, esimerkkipiirien teoriaosuuden sähkötekniikan sanat, joille kirjastosta löytyi määritelmä, linkitettiin kirjastoon. Näin käyttäjälle luotiin mahdollisuus esimerkkipiirejä opiskellessaan tarkistaa haluamansa määritelmä helposti kirjastosta. Työn tilaaja ei ehtinyt toimittaa teoriakirjaston materiaalia sovitussa ajassa, joten päätettiin että, HTML-tiedostoihin tulevien määritelmien lisääminen jää työn tilaajan vastuulle.

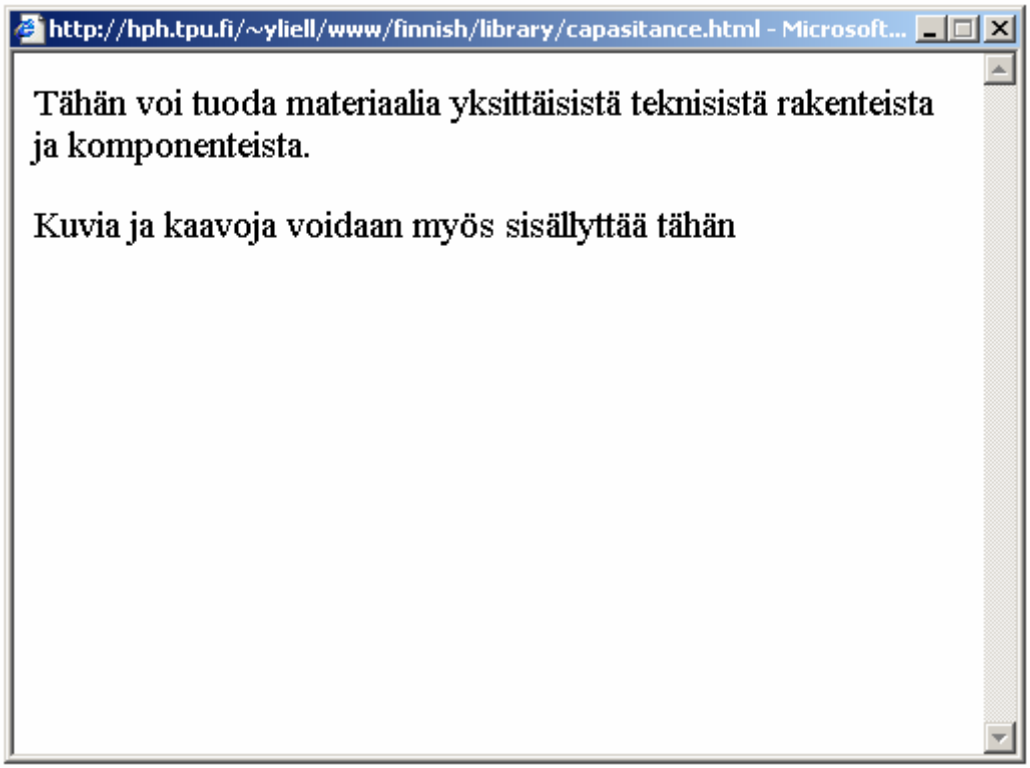
7.2 Kirjaston rakenne

Teoriakirjaston koon kasvu tulevaisuudessa huomioitiin jo alkuvaiheessa, jotta teoriakirjaston kehitys tulevaisuudessa ei vaatisi suuria muutoksia. Ensin kirjastosta tehtiin sisällysluettelona toimiva HTML-tiedosto, josta oli linkit teorian sisällään pitäviin HTML-tiedostoihin. Jokaiselle määritelmälle tehtiin oma HTML-tiedosto, jotta käyttäjä voisi teoriakirjaston sisällysluettelosta valita haluamansa aiheen. Määritelmät haluttiin pitää määritelmäkohtaisina kokonaisuuksina, joten siksi ne sijoitettiin eri sivuille (kuva 8).

Kirjaston kasvaessa tarvitaan teoriakokonaisuuksille oma sisällysluettelo ja edelleen kokonaisuuksien yksityiskohtaiset sisällysluettelot ja nämä kaikki täytyy. Tämän takia luotiin kielikohtaiset teoriakirjaston kansiot, joihin siirrettiin kaikki teoriakirjaston HTML-tiedostot, jotka pitävät sisällään yhden määritelmän. Teoriakirjaston sisällysluetteloiden HTML-tiedostot pidettiin TESTI-ohjelman kielikansioissa samalla tasolla tiedostorakenteessa, jotta näistä voidaan tehdä linkitys määritelmät sisältävään kansioon ilman tiedostopolkuun tehtäviä muutoksia. Tämän ansiosta myös esimerkkipiirien linkit teoriakirjastoon voidaan säilyttää muuttumattomina, vaikka teoriakirjaston sisällysluetteloita ja määritelmiä lisätään. Esimerkkipiirien linkit osoittavat suoraan teoriakirjaston kansioon, joten sisällysluetteloiden lisääminen ei vaikuta näihin linkkeihin. Teoriakirjaston tietomäärää voidaan kasvattaa ja tiedon hakua helpottaa ilman työlästä linkkien päivitystä.

7.3 Uudet ikkunat

Pop up –ikkunoiden ja JavaScriptin käyttöä TESTI-ohjelmassa harkittiin pitkään. TESTI-ohjelman tuli toimia kaikissa selaimissa ja olla kaikkien käytettävissä. Useat käyttäjä kuitenkin kytkevät JavaScriptin pois päältä ja vielä useammat käyttäjät vierastavat pop up –ikkunoita, jouduttuaan useasti Internetin pop up –ikkunamainosten uhriksi. Tämä huomioitiin jo kirjastoa suunnitellessa, koska teoriakirjaston käytön piti olla mahdollista kaikille käyttäjille. Päädyttiin ratkaisuun, jossa käytetään JavaScriptiä ja pop up –ikkunoita, mutta kirjaston selaus esimerkeistä on mahdollista myös ilman JavaScriptiä. Jos käyttäjä sallii pop up –ikkunoiden käytön, linkki aukeaa sopivaksi mitoitettuun ja vain välttämättömät ominaisuudet sisältävään ikkunaan ja on esillä käyttäjälle samaan aikaan teoria- ja simulointiosuuden kanssa. Mikäli käyttäjä on kieltänyt JavaScriptin käytön linkki aukeaa uuteen selaimen ikkunaan ja on käyttäjän itsensä säädettävissä. Pop up –ikkuna on nähtävissä kuvassa 8.



Kuva 8. Pop up –ikkuna, joka aukeaa esimerkkipiirin linkistä. Materiaali lisätään HTML-tiedostoihin jatkokehityksen yhteydessä.

Ratkaisussa käytettiin pop up –ikkunoita, koska TESTI-ohjelman tilaajan mielestä nämä olivat tässä kohtaa perusteltavissa käyttäjän näkymän arvokkaana lisätilana. Käyttäjän ei myöskään tarvitse keskeyttää esimerkkipiirin tarkastelua määritelmää tutkiakseen, vaan linkki aukeaa käyttäjän näkymään ja käyttäjä pystyy siirtämään pop up -ikkunan haluamaansa kohtaan. Ratkaisussa otettiin huomioon myös JavaScriptin kieltäminen, joten teoriakirjastoa voi esimerkkipiireistä selata myös JavaScrptin käytön selaimellaan kieltäneet käyttäjät. Jos käyttäjä on kokonaan kieltänyt uusien ikkunoiden avaamisen, teoriakirjaston selaaminen esimerkkipiireistä ei onnistu. Työ tilaajan kanssa sovittiin, että TESTI-ohjelmaan kirjoitetaan ohjeistus, jossa tämä kerrotaan. Jos käyttäjä ei tästä huolimatta halua sallia pop up –ikkunoita tai uusia ikkunoita, täytyy teoriakirjaston selaaminen tehdä valikon kirjastolinkin kautta.

8 Osoitinkuvat

Havainnollisuuden lisäämiseksi ohjelmaan haluttiin dynaamisesti piirrettyjä sähkötekniikan osoitinpiirroksia. Osoittimien piirtämiseen piti käyttää samoja käyttäjän antamia arvoja, kuin Gnuplot-ohjelman piirtämissä kuvaajissa, jotta käyttäjä voisi vertailla samaa ilmiötä useamman havainnollistavan kuvan avulla. Kuva piti tästä syystä saada käyttäjän näkymään samaan aikaan Gnuplot-ohjelman piirtämän kuvan kanssa.

8.1 Toteutustavan pohdinta

Osoitinpiirroksia varten tutkittiin ensin Internetissä saatavilla olevia grafiikanpiirto-ohjelmia, jotta voitaisiin hyödyntää valmista osoitinpiirroksia piirtävää ohjelmaa Gnuplot-ohjelman tavoin. Gnuplot-ohjelmaa tutkittiin myös, mutta keinoa piirtää osoitinpiirroksissa tarvittavia nuolia ei löydetty. Toiveena oli löytää yksinkertainen sovellus, joka Gnuplot-ohjelman tavoin ottaa vastaan syötteitä ja palauttaa halutun kuvaajan. Lisäksi ohjelman tuli olla käytettävissä ilman lisenssimaksuja, kuten työn tilaajan kanssa oli sovittu. Ohjelmalle syötteinä annettaisiin osoittimen alku- ja loppupisteen koordinaatit tai vaihtoehtoisesti osoittimen pituus sekä osoittimen ja koordinaatiston x-akselin välinen kulma. Internetistä löydetyistä grafiikanpiirto-ohjelmista yksikään ei täysin täyttänyt näitä vaatimuksia. Osoittimien piirron mahdollistavia grafiikanpiirto-ohjelmia löytyi kyllä, mutta ne olivat joko liian monimutkaisia yksinkertaiseen osoitinpiirrokseen tai lisenssin vaativia ohjelmia.

Seuraavaksi tutkittiin PHP:n valmiita luokkakirjastoja, joista löydettiin grafiikan piirtoon ja kuvan luomiseen tarkoitettuja kirjastoja. Osoitinpiirroksen osoittimien suuntien ja suuruuksien laskemiseksi tarvittaviin matemaattisiin laskutoimituksiin löydettiin myös keinot PHP:n Math-funktioista. Todettiin, että yksinkertaisin tapa toteuttaa osoitinkuvien piirtäminen, olisi kirjoittaa piirtämiseen ja kuvan luomiseen tarvittava koodi PHP:n funktioiden avulla.

8.2 Toteutus

Toteuttaminen tehtiin kahdessa vaiheessa. Ensin toteutettiin vaikeammaksi arvioitu piirtäminen, koska haluttiin varmistaa sen onnistuminen PHP:tä käyttäen. Tämän jälkeen siirryttiin osoittimien pituuksien laskemiseen ja arvojen välittämiseen kuvan piirtävälle PHP-tiedostolle.

8.2.1 Piirtäminen

Ensimmäiseksi lähdettiin liikkeelle viivan piirtämisestä, koska nuoli saadaan koostettua kolmesta viivasta. Löydettiin funktio, joka piirtää parametrina annettuun kuvaan viivan. Viiva piirretään siten, että alku- ja loppupiste sekä viivan väri ja kuva, johon viiva piirretään, annetaan sille parametreina. Tällä funktiolla piirretty viiva oli kuitenkin liian ohut näkymään selkeästi, koska se oli vain yhden kuvapisteen levyinen. Tämän sijaan käytettiin nelikulmiota, koska sillä saatiin piirrettyä kuvio, joka näyttää leveältä viivalta. Tähän käytettiin PHP:n funktiota, joka piirtää määriteltyyn kuvaan halutulla värillä piirretyn ja täytetyn monikulmion.

PHP-esimerkki:

```
imagefilledpolygon ( $picture, $array, 4, $color );
```

Funktiolle annetaan parametreiksi monikulmion kulmien koordinaatit taulukkona, joka on `$array` yllä olevassa esimerkissä /1/. Muut parametrit ovat monikulmion kulmien määrä ja monikulmion väri sekä kuva, johon monikulmio piirretään/1/. Tällä funktiolla voitiin piirtää halutun levyinen viiva antamalla sopivat koordinaatit, joten se todettiin sopivan osoitinpiirroksen viivaksi.

Osoitinpiirroksen nelikulmiosta muodostetun viivan piirtämiseksi tarvittiin alku- ja loppukoordinaatit, joiden avulla viivaa vastaava kuvio muodostetaan. Nämä pisteet oli saatava käyttäjän syötteistä lasketuista osoittimien pituuksista ja osoittimen pää oli piirrettävä nuolen muotoon. Viivan leveys ja väri haluttiin myös muuttujiksi, jotta havainnollisuutta voitaisiin lisätä näiden avulla. Viivan piirtämiseksi tehtiin funktio PHP:n esimerkkejä apuna käyttäen/1/. Funktio löytyy liitteestä 6.

PHP-esimerkki:

```
drawLine($picture,$start_x,$start_y,$end_x,$end_y,$colo
r,$width)
```

Funktiossa lasketaan viivalle leveys parametrina annetun leveysmuuttujan arvolla. Tämän jälkeen piirretään halutun leveyden mukainen viiva kutsumalla PHP:n monikulmion piirtävää funktiota ja antamalla sille parametreina lasketut nelikulmion kulmien pisteiden koordinaatit. Parametreina annettaan myös muut funktion vaatimat muuttujat, joita ovat kuvion väri ja kuva, johon kuvio piirretään. Kulmien määrä on TESTI-ohjelmaan piirrettävissä monikulmioissa aina neljä, joten se voitiin ilmoittaa vakiona monikulmion piirtävälle funktiolle. Kuvan ja värin muuttujia ei käsitelty leveän viivan piirtävässä funktiossa mitenkään, mutta ne tarvittiin PHP:n monikulmion piirtävälle funktiolle parametreiksi. Monikulmion piirtävää funktiota kutsutaan leveän viivan piirtävästä funktiosta, joten kuvan ja värin muuttujat täytyi tästä syystä välittää jo leveän viivan piirtävälle funktiolle.

Osoitinpiirroksesta tehtävään kuvaan piirretään ensimmäiseksi koordinaatisto, joka on sama kaikissa sähkötekniikan piireistä piirretyistä osoitinkuvissa. Koordinaatisto haluttiin kuvaan havainnollistamaan osoittimien negatiivisia ja positiivisia arvoja. Osoittimien piirtämiseksi käytettiin muuttujia, joihin käyttäjän syötteistä laskettavat arvot sijoitettaisiin (Liite 6).

PHP-esimerkki

```
drawLine($picture,10,100,$resistance,100,$red,0.5);
```

Esimerkissä parametreina annetaan ensin kuva, johon piirretään. Kaksi seuraavaa parametria kertovat alkupisteen koordinaatit, jotka kuvassa ovat piirretyn koordinaatiston origon koordinaatit. Kaksi seuraavaa parametria sisältävät loppupisteen koordinaatit. Muuttuja sisältää käyttäjän syötteistä lasketun arvon, joka on resistanssin yli olevan jännitteen suuruutta kuvaavan osoittimen pituus. Tuloksena on x-akselin suuntainen viiva, jonka pituus on muuttujan sisältämä arvo. Kaksi viimeistä parametria sisältävät värin ja halutun viivanleveyden.

Kuvapisteiden koordinaatit ovat eri koordinaatit, kuin funktiolla piirrettävän koordinaatiston koordinaatit. Kuvan x- ja y-koordinaatit ovat nollia kuvan vasemmassa yläkulmassa, kun taas piirretyn koordinaatiston x- ja y-koordinaatit ovat nollia kohdassa (10, 100) kuvan koordinaatteina. Tämä täytyi huomioida osoittimia piirrettäessä, jotta osoittimien alku- ja loppukoordinaatit saatiin vastaamaan piirrettyä koordinaatistoa. Tämä ratkaistiin siten, että origosta lähtevien osoittimien koordinaatit asetettiin alkamaan kohdasta (10, 100), y-akselilta lähtevien osoittimien muuttujiin lisättiin luku kymmenen ja y-akselista lähteviin osoittimiin lisättiin luku sata. Näillä lisäyksillä saatiin piirrettyä osoittimet oikein TESTI-ohjelman osoitinkuvan koordinaatistoon. Ratkaisu päti kaikkiin esimerkkipiireihin, joista osoitinpiirros voitiin tehdä, joten se todettiin käyväksi tähän tarkoitukseen.

Osoittimen suunnan ilmaisemiseksi tarvittiin nuolenpää, joka piirrettiin muuttujien avulla. Piirtämiseen käytettiin samaa funktiota, jolla osoittimien pituuden määräämä viiva piirrettiin.

PHP-esimerkki

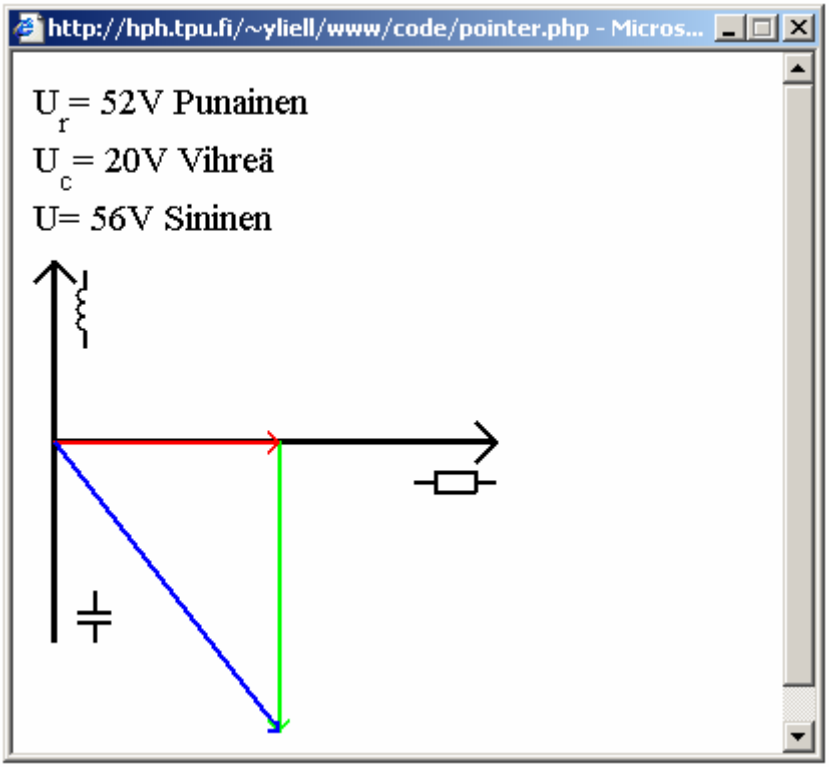
```
drawLine($picture,$resistance-5,95,$resistance,100,
$red,0.5);

drawLine($picture,$resistance-5,105,$resistance,100,
$red,0.5);
```

Parametreina annettavat koordinaatit määriteltiin muuttujan avulla. Molempien viivojen päätepiste, eli nuolen kärki, on muuttujan arvo x-akselilla ja y-akselilla arvo on sata, koska se on osoitinkuvan y-koordinaatti TESTI-ohjelmasta piirretyn koordinaatiston x-akselille. Alkupisteen x-koordinaatit ovat osoittimen loppupisteistä viisi yksikköä origoa kohti ja y-arvoista toinen on viisi yksikköä ylöspäin ja toinen viisi yksikköä alaspäin. Nuoli saatiin näin piirrettyä muuttujan arvon avulla, jolloin nuolenpää voidaan piirtää osoittimelle, jonka tiedetään kulkevan aina x-akselia pitkin. Muut osoittimet piirretään vastaavasti laskettuja arvoja käyttäen ja nuolenpää piirretään osoittimien arvoja hyväksi käyttäen. Tämä ratkaisu toimi kaikille TESTI-ohjelmassa piirrettäville osoitinpiirroksille. Osoitinpiirroksen tekevää PHP-tiedostoa voi tutkia liitteestä 6.

Osoitinpiirroksen kuva haluttiin näkymään yhtä aikaa Gnuplot-ohjelman tekemän kuvaajan kanssa, jotta käyttäjä voisi vertailla erilaisia simulaatioita samoilla arvoilla. Osoitinpiirrosta ei kuitenkaan voitu sijoittaa piirikaavion paikalle, koska käyttäjän piti voida vertailla piirikaaviota siitä tehtyihin simulaatioihin. Teoriaosuuden paikalle osoitinpiirrosta ei myöskään haluttu, koska se haluttiin pitää omana kokonaisuutenaan. Paikan ongelma ratkaistiin tekemällä osoitinpiirroksesta linkki, joka aukeaa pop up –ikkunaan. Osoitinpiirros voitiin tämän ansiosta siirtää haluttuun kohtaan näkymässä ja vertailu piirikaavion ja Gnuplot-ohjelman kuvaajien kanssa oli mahdollista. Linkistä aukeaa JavaScriptin avulla pop up -ikkuna, johon osoitinpiirros ja jännitteiden todelliset arvot sijoitetaan (kuva 9). Mikäli käyttäjä on estänyt JavaScriptin käytön, osoitinpiirros aukeaa uuteen ikkunaan.

Osoitinpiirros saatiin piirrettyä suunnitellusti ja sitä voidaan soveltaa myös TESTI-ohjelmaan tulevaisuudessa lisättäviin esimerkkipiirien osoitinpiirroksiin.



Kuva 9. Osoitinkuva.

8.2.2 Arvojen laskeminen

Osoittimien pituudet piti laskea TESTI-ohjelmassa, koska Gnuplot-ohjelma ei palauta jännitteiden suuruuksia, vaan ainoastaan kuvaajat. Käyttäjän syötteet otettiin vastaan ja tarkistettiin esimerkkipiirien Gnuplot-ohjelman käyttöliitymien PHP-tiedostoissa. Laskenta suoritettiin käyttäjän antamat arvot vastaanottavassa PHP-tiedostossa, koska laskenta haluttiin suorittaa ennen Gnuplot-ohjelman komentotiedoston kirjoittamista. Laskenta olisi voitu suorittaa myös Gnuplot-ohjelman komentotiedoston kirjoittavassa PHP-tiedostossa, mutta näiden tiedostojen selkeyden säilyttämiseksi laskenta tehtiin käyttöliittymän PHP-tiedostossa. Sähkötekniikan kaavojen avulla saadaan laskettua halutut jännitteiden arvot, jotka sijoitetaan istunnon muuttujiin. Lasketut arvot ovat näiden muuttujien avulla käytettävissä PHP-tiedostossa, joka piirtää osoitinpiirroksen ja PHP-tiedostossa, joka luo HTML-sivun, johon lisätään osoitinpiirros ja kokonaisluvuiksi pyöristetyt jännitteiden arvot (Kuva 9).

Havainnollisuuden säilyttämiseksi kaikissa tapauksissa päätettiin, että esimerkkipiiristä piirrettävissä osoitinpiirroksissa piirin kokonaisjännitettä kuvaavan osoittimen pituus on vakio ja muiden osoittimien pituudet suhteutetaan tämän osoittimen pituuteen. Osoitinpiirros voidaan tällöin sijoittaa aina saman suuruiseen koordinaatistoon ja kuvasta on nähtävissä osoittimien suhteet, vaikka kokonaisjännitteen arvot vaihtelevat yhdestä sataan volttia. Osoitinpiirroksen piirtävässä PHP-tiedostossa käytetään suhteutettuja osoittimien pituuksia, jotka lasketaan jännitteiden suuruksien yhteydessä. Osoittimien pituuksien laskenta PHP:llä löytyy liitteestä 5.

9 Yhteenveto

Ohjelman muokkaaminen kokeellisesta versiosta opetuskäyttöön soveltuvaksi versioksi oli työlästä. Lähtötilanteessa toimiva koodi oli suurelta osin käyttökelpoista, mutta vaati kuitenkin pientä muokkausta. Työtä olisi nopeuttanut kunnollinen ohjelmakoodin kommentointi, joka ohjelmointityön siirtyessä tekijältä toiselle on käytännöllinen tapa kertoa, mitä tekijä on missäkin kohtaa tehnyt tai virheiden löytyessä, miten koodin olisi pitänyt toimia. Tavallisesta käytännöstä poiketen tällä ohjelmistoprojektilla ei ollut dokumentoituna vesiputousmallin mukaisia dokumentteja, oli vain yksi dokumentti, jota täydennettiin pääasiassa ohjelmakoodin valmistuessa. Ohjelmoijan kannalta oli hankalaa, kun määrittely voi muuttua milloin tahansa.

TESTI-ohjelmaan toteutettiin suunnitellut muutokset ja se toimii niin kuin suunniteltiin. Käyttäjäryhmät toteutettiin ja käyttäjien hallinta muokattiin vaatimusten mukaiseksi. Kielivalinnan takia TESTI-ohjelmaan piti toteuttaa suuria muutoksia, jotta kielenkääntäminen saatiin tehtyä vaatimusten mukaiseksi. TESTI-ohjelman tiedostorakenteen ja aloitussivun muokkaus tehtiin myös kielen valintaa muokattaessa. Kielivalinta toteutettiin vaatimusten mukaiseksi ja kielenkääntäminen voitiin toteuttaa HTML-kieltä osaavalla kielenkääntäjällä. Teoriakirjaston pohja luotiin TESTI-ohjelmalle, mutta sinne lisättävän materiaalin viipymisen takia sisältöä ei tämän työn aikana voitu sinne lisätä. Sisällön tuottaminen oli työn tilaajan vastuulla ja sovittiin, että työn tilaaja lisää sisällön TESTI-ohjelman teoriakirjaston HTML-tiedostoihin työn tekijän antamien ohjeiden mukaan. Osoitinpiirrosten lisääminen TESTI-ohjelmaan haluttiin ensin tehdä valmista grafiikanpiirto-ohjelmaa hyödyntäen. Sopivaa ohjelmaa ei kuitenkaan löytynyt ja osoitinpiirrokset päätettiin toteuttaa itse. PHP:n dokumentaation ja esimerkkien avulla osoitinpiirrokset saatiin toteutettua. Osoitinpiirrokset toimivat, kuten suunniteltiin ja ne ovat helposti muokattavissa myös jatkokehityksen tarpeisiin.

Työn aikataulua jouduttiin kertaalleen uudistamaan. Tämä johtui tutkintotyön virallisen aloitusluvan viivästymisestä ja tutkintotyön tilaajan kanssa tehdystä uudesta työsopimuksesta, joka ei ollut tiedossa tutkintotyön ensimmäistä aikataulua tehtäessä. Lisäksi avainhenkilöiden tavoittaminen ja aikataulujen yhteensovittaminen oli haasteellista. Työ valmistui vähän aikataulustaan jäljessä, mikä johtui edellä mainituista syistä. TESTI-ohjelman tilaajalle tämä ei kuitenkaan ollut ongelma, koska opetuskäyttöön TESTI-ohjelma on tarkoitus ottaa vasta tulevana syksynä.

10 Jatkokehitys

TESTI-ohjelman päätavoite jatkokehitykselle on tämän hetken suunnitelmien mukaan opetusmateriaalin määrän kasvattaminen. TESTI-ohjelman opiskelukielen kääntäminen uusille kielille ja kansainvälisten yhteistyökumppaneiden löytäminen ohjelman jatkokehityskehitystyöhön ovat myös lähitulevaisuuden tavoitteita. Tätä tutkintotyötä tehdessä syntyi myös jatkokehitysajatuksia, joita ei aikataulujen puitteissa voitu tässä työssä toteuttaa.

TESTI-ohjelman toimintaa opetuskäytössä ei ole vielä testattu. Opetuskäytön kokeilun jälkeen tiedetään paremmin, mitkä TESTI-ohjelman osat ovat

opetuskäytön kannalta hyödyllisiä ja mitkä eivät. Opiskelijan kannalta opiskelumateriaalin lisäys olisi hyödyllisintä tässä kohtaa TESTI-ohjelman kehitystä. Harjoitustehtävät olisivat myös hyödyllisiä opiskelijan oppimisen kannalta. TESTI-ohjelmaan voisi kehittää harjoitustehtäviä, joita opiskelija voi tehdä teoria- ja simulointiosuuteen tutustumisen jälkeen. Harjoitukset voisivat pitää sisällään teoriakysymyksiä, joihin olisi vastausvaihtoehdot. Tämä olisi yksinkertaista toteuttaa HTML-lomakkeiden avulla ja lomakkeiden tarkistaminen onnistuisi PHP:n avulla helposti. Harjoitustehtäviin voisi myös kuulua laskutehtäviä, joista TESTI-ohjelma osaa kertoa oikeat vastaukset tai tehtävien vastaukset voisi TESTI-ohjelman kautta lähettää sähköisesti valmiilla pohjalla opettajan tarkastettavaksi. TESTI-ohjelman omien sähkötekniikan kaavoista tehtyjen simulaatioiden mahdollisuutta, jonka kokeiluversio piti sisällään, voisi myös kehittää.

Kehityksessä voisi ottaa huomioon opettajien ja oppilaiden erilaiset tarpeet sähkötekniikan ilmiöiden simulointiin. Opettajille voisi tehdä mahdollisuuden lisätä TESTI-ohjelmaan pysyviä esimerkkipiirejä simulointimahdollisuuksineen. Oppilaille voisi kehittää mahdollisuuden simuloida omia esimerkkejään, vaikka sähkötekniikan kurssikirjasta. Näitä simulointeja ei säilytettäisi, vaan opiskelija määrittelisi piirit ja kaavat aina uudestaan, jotta epäonnistuneita ja päällekkäisiä simulaatioita ei säilytettäisi turhaan palvelimella. Toinen mahdollisuus oppilaiden omiin simulointeihin olisi oppilaskohtainen tietopankki, joka sisältäisi oppilaan omat piirit. Tämä voitaisiin toteuttaa niin, että tietopankin sisältö olisi oppilaan omalla tietokoneella ja TESTI-ohjelmasta voitaisiin käyttää tätä materiaalia. Opettajillekin tietopankki toisi mahdollisuuksia opetuksen havainnollistamiseen.

Lähteet

1 PHP:n kotisivut [web-sivu] [viitattu 20.4.2006] Saatavissa: http://php.net

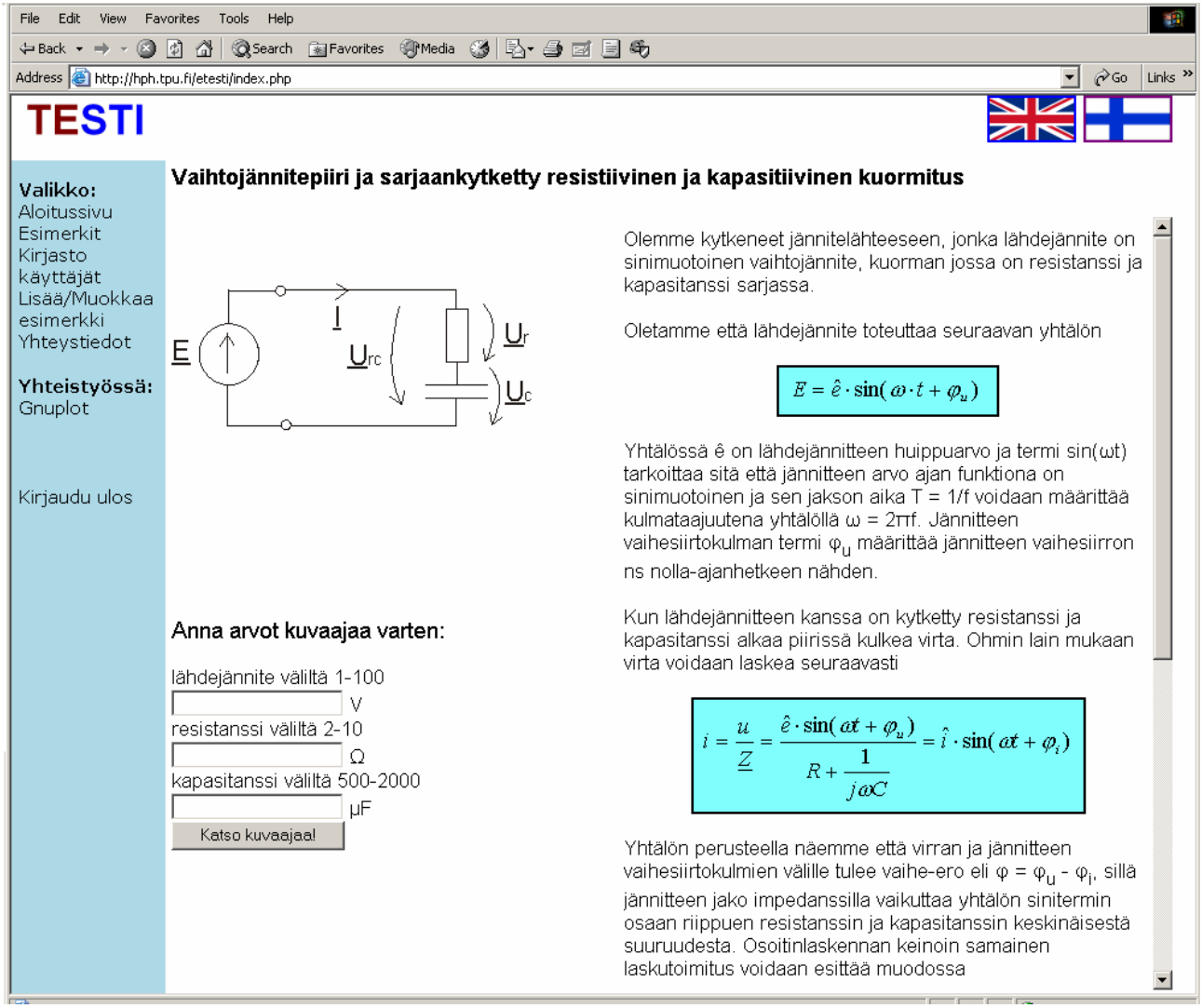
2 w3schools kotisivut [www-sivu]. [viitattu 20.4.2006] Saatavissa: http://www.w3schools.com

3 MySQL:n kotisivut [www-sivu]. [viitattu 31.3.2006] Saatavissa: http://www.mysql.com

4 Gnuplot-ohjelman kotisivut [www-sivu]. [viitattu 31.3.2006] Saatavissa: http://www.gnuplot.info

5 Virtuaaliammattikorkeakoulun kotisivut [www-sivu]. [viitattu 31.3.2006] Saatavissa: http://www.virtuaaliamk.fi

TESTI-ohjelman koko näkymä esimerkkipiiristä.



Gnuplot-ohjelman komentotiedoston kirjoittava PHP-tiedosto

```
<?php
$lang=$_SESSION['lang'];
$id=$_SESSION['id'];
echo '<form method="POST" formaction="acclrinterface.php">';
include("../../$lang/tryagain.html");
strval($id);
$id=$id ."gnuplot.txt";
$file=fopen($id ,'w'); //users file
fwrite($file, "# AC-circuit analyse\r\n#\r\n");
fwrite($file, "# Prameters from simulation circuit\r\n#\r\n");
fwrite($file, "# DRAWING ONE CURVE\r\n#\r\n");
fwrite($file, "# Resistance R, capasitance C and inductance L in series and source
Voltage E\r\n#\r\n");
fwrite($file, "#\r\n# IF ONE IS NOT USED (C OR L) IT IS NEGLEGTED BY USING VERY SMALL
VALUE FOR L AND VERY BIG VALUE FOR C.\r\n#\r\n#\r\n");
fwrite($file, "R(Td)="); fwrite($file, $_SESSION["resistance"]); fwrite($file, " \r\n");
fwrite($file, "E(Td)="); fwrite($file, $_SESSION['voltage']); fwrite($file, " \r\n");
fwrite($file, "C(Td)="); fwrite($file, $_SESSION['capasitance']); fwrite($file, "
\r\n");
fwrite($file, "L(Td)="); fwrite($file, $_SESSION['inductance']); fwrite($file, " \r\n");
fwrite($file, "F(Td)=50\r\n");
fwrite($file, "#\r\n# Current equation after the switch is closed\r\n#\r\n");
fwrite($file, "Alfa(Td)= atan(((2*3.14159*L(Td)*F(Td))-
1/(2*3.14159*C(Td)*F(Td)))/R(Td))\r\n");
fwrite($file, "Abs(Td)= (R(Td)**2+(2*3.14159*F(Td)*L(Td)-
1/(2*3.14159*F(Td)*C(Td)))**2)**(0.50)\r\n");
fwrite($file, "I1(Td)= E(Td)/Abs(Td)*sin(2*Td+Alfa(Td))\r\n");
fwrite($file, "I2(Td)= E(Td)/Abs(Td)*sin(2*Td+Alfa(Td)-3.14159/2)\r\n");
fwrite($file, "I3(Td)= E(Td)/Abs(Td)*sin(2*Td+Alfa(Td)+3.14159/2)\r\n");
fwrite($file, "#\r\n#\r\n# Voltage equation after the switch is closed\r\n#\r\n");
fwrite($file, "U1(Td)= E(Td)*sin(2*Td)\r\n");
fwrite($file, "U2(Td)=I1(Td)*R(Td)\r\n");
fwrite($file, "U3(Td)=I2(Td)*1/(2*3.14159*C(Td)*F(Td))\r\n");
fwrite($file, "U4(Td)=I3(Td)*2*3.14159*L(Td)*F(Td) \r\n#\r\n#\r\n");
fwrite($file, "# Text and grid definations\r\n#\r\n");
fwrite($file, "set terminal png size 350,280 \r\n");
fwrite($file, "set output \""); fwrite($file, $_SESSION['id']); fwrite($file,
"diag.png\"\r\n");
fwrite($file, "set dummy Td\r\n");
fwrite($file, "unset grid\r\n");
fwrite($file, "unset key\r\n");
fwrite($file, "set grid x y\r\n");
fwrite($file, "set offsets 0, 0, 0, 0\r\n");
fwrite($file, "set title \"resistive, inductive and capasitive load in series\"\r\n");
fwrite($file, "set xlabel \"Time (sec)\"\r\n");
fwrite($file, "set xrange [0 : 10]\r\n");
fwrite($file, "set ylabel \"Circuit current (A), Voltages (V)\"\r\n");
fwrite($file, "set yrange [-");fwrite($file, $x*1.1);fwrite($file, " : ");fwrite($file,
$x*1.1);fwrite($file, "]\r\n");
fwrite($file, "plot I1(Td),U1(Td),U2(Td),U3(Td),U4(Td)\r\n");
fwrite($file, "#plot I1(Td), I2(Td), U1(Td), Abs(Td)\r\n#\r\n");
fwrite($file, "set ytics nomirror\r\n");
fwrite($file, "set ytics\r\n");
fwrite($file, "set tics out\r\n");
fwrite($file, "set autoscale y\r\n");
fwrite($file, "set autoscale x\r\n");
fwrite($file, "unset label\r\n");

fwrite($file, "reset\r\n");
exec("/usr/bin/gnuplot $id");
?>
```

Gnuplot-ohjelman komentotiedosto

```
# AC-circuit analyse
#
# Prameters from simulation circuit
#
# DRAWING ONE CURVE
#
# Resistance R, capasitance C and inductance L in series and source Voltage E
#
#
# IF ONE IS NOT USED (C OR L) IT IS NEGLEGTED BY USING VERY SMALL VALUE FOR L AND VERY
# BIG VALUE FOR C.
#
#
R(Td)=5
E(Td)=55
C(Td)=0.000666
L(Td)=0.044
F(Td)=50
#
# Current equation after the switch is closed
#
Alfa(Td)= atan(((2*3.14159*L(Td)*F(Td))-1/(2*3.14159*C(Td)*F(Td)))/R(Td))
Abs(Td)= (R(Td)**2+(2*3.14159*F(Td)*L(Td)-1/(2*3.14159*F(Td)*C(Td)))**2)**(0.50)
I1(Td)= E(Td)/Abs(Td)*sin(2*Td+Alfa(Td))
I2(Td)= E(Td)/Abs(Td)*sin(2*Td+Alfa(Td)-3.14159/2)
I3(Td)= E(Td)/Abs(Td)*sin(2*Td+Alfa(Td)+3.14159/2)
#
#
# Voltage equation after the switch is closed
#
U1(Td)= E(Td)*sin(2*Td)
U2(Td)=I1(Td)*R(Td)
U3(Td)=I2(Td)*1/(2*3.14159*C(Td)*F(Td))
U4(Td)=I3(Td)*2*3.14159*L(Td)*F(Td)
#
#
# Text and grid defination
#
set terminal png size 350,280
set output "1diag.png"
set dummy Td
unset grid
unset key
set grid x y
set offsets 0, 0, 0, 0
set title "resistive, inductive and capasitive load in series"
set xlabel "Time (sec)"
set xrange [0 : 10]
set ylabel "Circuit current (A), Voltages (V)"
set yrange [-60.5 : 60.5]
plot I1(Td),U1(Td),U2(Td),U3(Td),U4(Td)
#plot I1(Td), I2(Td), U1(Td), Abs(Td)
#
set ytics nomirror
set ytics
set tics out
set autoscale y
set autoscale x
unset label
reset
```

Aloitussivun PHP-tiedosto

```
<?php
session_start();
if(isset($_GET['logout']))        //ensimmäinen tarkistus
{
     $_SESSION['username']='';
     $_SESSION['usergroup']='';
     $_SESSION['lang']='';
}
if(isset($_POST['username']) && isset($_POST['userpass'])) //toinen tarkistus
{
     $username=$_POST['username'];
     $userpass=$_POST['userpass'];
     include ("code/sqlsettings.php");
     mysql_connect($serveraddress,$user,$password);
     @mysql_select_db($database) or die( "Unable to select database");
$query = "SELECT id, username, userpass, usergroup FROM users WHERE
username='$username' AND userpass='$userpass'";
     $result=mysql_query($query);
     $num=mysql_num_rows($result);
     mysql_close();
     if($num==1)
     {
            $row=mysql_fetch_assoc($result);
            $_SESSION['id']=$row['id'];
            $_SESSION['username']=$row['username'];
            $_SESSION['usergroup']=$row['usergroup'];
     }
     else
     {
            echo("Sorry, this login is invalid.");
     }
}
if(isset($_GET['lang']))          //kolmas tarkistus
{
     $_SESSION['lang']=$_GET['lang'];
}
if(isset($_SESSION['username']) && $_SESSION['username']!= '') //neljäs tarkistus
{
     $group=$_SESSION['usergroup'];
     $lang=$_SESSION['lang'];
     ?>
     <html>
     <head>
     <title>TESTI - Index (English Language)</title>
     </head>
     <frameset rows="7%,93%" frameborder=0>
     <frame src="<?php echo $lang;?>/languageswitcher.html" name="switcher">
                    <frameset cols="13%,87%">
<frame src="<?php echo  $lang;?>/left<?php echo $group; ?>.html" name="left">
            <frame src="<?php echo $lang;?>/welcome.html" name="right">
                    </frameset>
     </frameset></html><?php
     }
else if(isset($_SESSION['lang']) && $_SESSION['lang']!= '') //viides tarkistus
{
     $lang=$_SESSION['lang'];
     include("$lang/adduserloginform.html");
}
else
{?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">
<html>
<head>
<title>TESTI - A Project In Progress</title>
</head><body><center>
<img src="images/testialphalogo.png"><p>
<a href="?lang=english"><img src="images/englishsmall.png"></a>
<a href="?lang=finnish"><img src="images/finnishsmall.png"></a>
</center></body></html><?php}?>
```

Gnuplot-ohjelman käyttöliittymän PHP-tiedosto

```
<?php
$lang=$_SESSION['lang'];

if(!isset($_POST['x']) && !isset($_POST['z'] && !isset($_POST['y'])))
{
   include("../../$lang/accrcircuit/insertvalues.html");
}
else
{
   $error=false;
   $x=$_POST['x'];
   $y=$_POST['y'];
   $z=$_POST['z'];

   if(!is_numeric($x) || !is_numeric($z) || !is_numeric($y) )  //käyttäjän syötteiden
          $error=true;                                        //Tarkastus

   else if($x==0 || $z==0 || $y==0)
          $error=true;

   if(($x < '1') || ($x > '100'))
          $error=true;

   if(($y < '2') || ($y > '10'))
          $error=true;

   if(($z < '500') || ($z > '2000'))
          $error=true;

   if ($error)
          include('error2.html');

   $_SESSION['resistance']=$y;                  //Sijoituksest istunnon muuttujiin
   $_SESSION['capasitance']=$z*0.000001;        //komentotiedoston kirjoittmista varten
   $_SESSION['voltage']=$x;

   $jxc=1/(2*3.1415926*50*$_SESSION['capasitance']);  //Osoitinpiirroksen arvojen
   $impedance=sqrt(pow($y, 2)+pow($jxc, 2));          // laskeminen
   $i=($x / $impedance);
   $ur= ($y * $i);
   $uc= ($jxc * $i);
   $relative=200/$x;
   $_SESSION['trueUr']=$ur;                            //sijoitukset istunnon
   $_SESSION['trueUc']=$uc;                            // muuttujiin
   $_SESSION['drawresistance']=$ur*$relative;          // osoitinpiirrosta varten
   $_SESSION['drawcapasitance']=$uc*$relative;

   else                                                //Komentotiedoston kirjoittavan
   {                                                   // PHP-tiedoston suoritus
          include('accrprosessor.php');
          echo '<a href="../pointer.php" target="ikkuna"
          onClick="ikkunanAvaaja()">Osoitinkuva</a> ';
          echo '<img src="lowdiagram.php">';
   }
}
?>
```

Osoitinpiirroksen PHP-tiedosto

```
<?php
session_start();
header("Content-type: image/png");

function drawLine($picture,$start_x,$start_y,$end_x,$end_y,$color,$width)
{
   $angle=(atan2(($start_y - $end_y),($end_x - $start_x)));

   $distance_x=$width*(sin($angle));
   $distance_y=$width*(cos($angle));
                                                        //drawLine() on
   $polygon_x1=ceil(($start_x + $distance_x));          //leveän viivan
   $polygon_y1=ceil(($start_y + $distance_y));          //piirtävä
   $polygon_x2=ceil(($end_x + $distance_x));            //funktio
   $polygon_y2=ceil(($end_y + $distance_y));
   $polygon_x3=ceil(($end_x - $distance_x));
   $polygon_y3=ceil(($end_y - $distance_y));
   $polygon_x4=ceil(($start_x - $distance_x));
   $polygon_y4=ceil(($start_y - $distance_y));

   $array=array(0=>$polygon_x1,$polygon_y1,$polygon_x2,$polygon_y2,
   $polygon_x3,$polygon_y3,$polygon_x4,$polygon_y4);

   imagefilledpolygon ( $picture, $array, 4, $color );
}
                                                        //käyttäjäkohtaiset arvot
$resistance=$_SESSION['drawresistance']+10;             //ja sovitus
piirrettävään
$capasitance=$_SESSION['drawcapasitance']+100;          //koordinaatistoon
$picture = ImageCreate (250, 250) or die("Cannot Initialize new GD image stream
");
$backgroundcolor = ImageColorAllocate ($picture, 255, 255, 255);

$red = ImageColorAllocate($picture, 255, 0, 0);
$blue = ImageColorAllocate($picture, 0, 0, 255);
$green = ImageColorAllocate($picture, 0, 255, 0);       //värien
$black = ImageColorAllocate($picture, 0, 0, 0);                 //määritys

drawLine($picture, 10, 10, 10, 200,$black,1);
drawLine($picture, 10, 100, 230, 100,$black,1);
drawLine($picture, 0, 20, 10, 10,$black,1);                     //koordinaatisto
drawLine($picture, 20, 20, 10, 10,$black,1);
drawLine($picture, 220, 90, 230, 100,$black,1);
drawLine($picture, 220, 110, 230, 100,$black,1);
                                                        //osoittimet

drawLine($picture, 10, 100, $resistance, 100, $red,0.5);
drawLine($picture, $resistance, 100, $resistance, $capasitance, $green,0.5);
drawLine($picture, 10, 100, $resistance, $capasitance, $blue,0.5);

drawLine($picture, $resistance-5, 95, $resistance, 100, $red,0.5);
     //nuolenpäät
drawLine($picture, $resistance-5, 105, $resistance, 100, $red,0.5);
drawLine($picture, $resistance+5, $capacitance-5, $resistance, $capasitance,
$green,0.5);
drawLine($picture, $resistance-5, $capacitance-5, $resistance, $capasitance,
$green,0.5);
drawLine($picture, $resistance-5, $capasitance, $resistance, $capasitance,
$blue,0.5);
drawLine($picture, $resistance-1, $capacitance-5, $resistance-1, $capasitance,
$blue,0.5);

imagepng($picture);
ImageDestroy($picture);

?>
```