Opinnäytetyö (AMK)

Tietojenkäsittely

Tietojärjestelmät

2012

Jarkko Hakala

AJANVARAUSJÄRJESTELMÄN SUUNNITTELU JA TOTEUTUS OPPILAITOSKÄYTTÖÖN

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely | Tietojärjestelmät

Maaliskuu 2012 | 81 sivua

Ohjaaja Anne Jumppanen

Jarkko Hakala

AJANVARAUSJÄRJESTELMÄN SUUNNITTELU JA TOTEUTUS OPPILAITOSKÄYTTÖÖN

Tämän opinnäytetyön aiheena on Internetissä toimivan ajanvarausjärjestelmän suunnittelu ja toteutus oppilaitoskäyttöön. Projekti aloitettiin haastattelemalla ohjelman tilaajaa Turun ammattikorkeakoulun Lemminkäisenkadun toimipisteen harjoittelun lehtori Kari Juhalaa ohjelman vaatimuksista. Ohjelman suunnittelu ja toteutus aloitettiin tyhjästä, sillä Kari Juhalalla ei ollut aikaisempaa tietokoneella toimivaa ajanvarausjärjestelmää. Työn tavoitteena oli luoda mahdollisimman helppokäyttöinen ja monipuolinen Internetissä toimiva ajanvarausjärjestelmä.

Ajanvarausjärjestelmän rakentamiseen käytettiin yleisiä www-tekniikoita, kuten HTML, PHP, CSS, JavaScript. PHP-ohjelmointikieli on suosittu Internet-sivujen toteutukseen tarkoitettu ohjelmointikieli ja sen vahvuuksia ovat monipuolisuus ja helppokäyttöisyys. Ohjelman tekemisessä käytettiin protoilumallia, joka mahdollistaa ohjelman jakamisen pienempiin osiin. Pienempiä ohjelman osia on helppo demonstroida ohjelman tilaajalle.

Ajanvaraus jakautui kahteen osaan: itse ajanvaraukseen ja ajanvarauksen hallintaan. Ajanvaraus sisältää ajanvarauksen lisäksi uutisien lukemisen ja aikojen peruuttamisen. Ajanvarauksen hallinnassa hallitaan ohjelman eri toimintoja, kuten aikoja, varauksia, uutisia ja tietokannan tauluja. Ohjelman tietojen tallentamiseen käytetään tietokantoja.

Opinnäytteessä esitelty ohjelman suunnittelu ja prototyyppien tekeminen onnistui hyvin. Työn tuloksena saatiin tehtyä toimiva prototyyppi ajanvarauksesta ja sen hallinnasta, mutta ohjelmaa ei saatu kokonaan valmiiksi, sillä testaus jäi osittain tekemättä.

ASIASANAT:

WWW-ohjelmointi, HTML, PHP, CSS, JavaScript

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Information Systems

March 2012 | 81 pages

Instructor Anne Jumppanen

Jarkko Hakala

DESIGN AND IMPLEMENTATION OF AN APPOINTMENT BOOKING SYSTEM FOR EDUCATIONAL INSTITUTION USE

The objectives of this thesis were to design and implement an internet based appointment system. The study was assigned by Turku University of Applied Sciences lecturer, Kari Juhala. The implementation of the site was done by PHP, HTML and CSS, which are the most widely used web design languages. The main objective and challenge was to design and make a website that is easy to use and maintain.

The project was implemented using prototype model. The Prototyping Model is a systems development method. The prototype purpose is to introduce the features of software or the user interface and its gives engineers and designers the possibility to explore design alternatives.

The final project started with planning the website along with the client in meetings and email discussions. The appointment system was divided into two separate sections: appointment system and its management.

As a result of this thesis I managed to create a fully working prototype of an appointment system. The finalized program was not completed during the thesis, because there wasn't enough time to do Software testing.

KEYWORDS:

WWW-programming, HTML, PHP, CSS, JavaScript

SISÄLTÖ

LIITTEET

KUVAT

KUVIOT

1 JOHDANTO

Tämän opinnäytetyön toimeksiantona on suunnitella ja toteuttaa Internetissä toimiva ajanvarausjärjestelmä. Ajanvaraus tulee Turun ammattikorkeakoulun Lemminkäisenkadun toimipisteen harjoittelun lehtori Kari Juhalan käyttöön. Kari Juhala tapaa työssään erittäin paljon ihmisiä, joten hän tarvitsee mahdollisimman nopean ja helpon tavan järjestää tapaamiset. Aikaisemmin tapaamiset on merkitty hänen työhuoneensa oven vieressä olevalle lapulle, johon on vapaan ajan kohdalle kirjoitettu oma nimi, tai ilmoitettu tapaaminen sähköpostilla. Molemmat tavat ovat erittäin hankalia ylläpitää, joten Internetissä toimiva ajanvaraus nopeuttaa ja helpottaa ajanvarauksen ylläpitämistä.

Opinnäytteessä selvitetään ajanvarauksen suunnittelussa ja tekemisessä tarvittavia Internet-tekniikoita, kuten PHP, HTML, CSS ja JavaScript. Ohjelmistotuotannosta kerron protoilumallista, jota käytin ohjelman tekemiseen. Tutkin myös valmiita ajanvarausohjelmia, joista sain hyviä ideoita omaan ohjelmaani ja miten erilaiset toiminnot kannattaa toteuttaa. Opinnäytteessä on myös koodiesimerkki tekemästäni tietokantaluokasta, jonka avulla käsittelen tietokannan tietoja.

Aiheen valitsin siksi, että järjestelmää tarvitaan oikeasti ja sen tekeminen vaikutti mielenkiintoiselta. Olen myös kiinnostunut WWW-sovelluksissa käytetyistä tekniikoista. Tavoitteena on kasvattaa tietämystäni niin web-ohjelmoinnista kuin ohjelman suunnittelusta.

2 OHJELMISTOTUOTANTO

Termiä ohjelmistotuotanto, software engineering, käytettiin ensimmäisen kerran jo vuonna 1968 käydyssä NATO-konferenssissa. Tuolloin keskusteltiin tietokoneohjelmien tekemisessä tapahtuvista ongelmista. Ongelmat olivat aivan samanlaisia tuolloin kuin nykyäänkin ohjelmistoprojekteissa. Kolme suurinta ongelmaa ohjelman tekemisessä tuohon aikaan oli:

1. Aikatauluongelmat: Tarkkoja aikatauluja ei pystytty tekemään, tai ne eivät olleet realistisia.
2. Epäluotettavat ohjelmat: Ohjelmissa oli ohjelmointivirheitä ja ne toimivat virheellisesti.
3. Hinnan arviointi: Ohjelmien todellisia hintoja ei pystytty kovinkaan tarkkaan ennustamaan. Ongelmana oli ohjelmien valmistumisen venyminen, tällöin myös ohjelman hinnan arvioiminen vaikeaa. (Sommerville 2011, 5)

Edellä mainituiden ongelmien johdosta piti kehittää uusia tekniikoita ja menetelmiä, jotta ongelmat saatiin korjattua. Uusia tekniikoita ja menetelmiä keksittiin vuosien 1970 ja 1980 välisenä aikana merkittävästi. Noina aikoina on keksitty mm. strukturoitu ohjelmointi ja olio-ohjelmointi, jotka helpottivat mm. suurien ohjelmien suunnittelua ja toteutusta. (Sommerville 2011, 5)

Englanninkielinen termi Software Engineering on yleensä suomennettu joko ohjelmistotekniikaksi tai ohjelmistotuotannoksi. Ohjelmistotekniikka ei ole kovin onnistunut käännös, sillä se voidaan helposti sekoittaa termiin ohjelmointitekniikka, joka käsittää vain pienen osan ohjelmistotekniikan osa-alueen. Termin ohjelmistotuotanto voidaan vapaasti tulkita tarkoittavan ohjelmistotyötä, jonka tuloksena syntyvät ohjelmat täyttävät tilaajan kohtuulliset vaatimukset ja odotukset, valmistuvat ajallaan ja kustannusarvioiden puitteissa. (Haikala 2004, 16)

Ohjelmistotuotanto on siis systemaattista työtä, joka tarjoaa välineet tehdä erilaisia ohjelmia, sillä yhtä universaalia menetelmää tai tapaa ohjelmien tekemiseen ei ole. Onkin projektikohtaista minkälaisia ohjelmistotuotannon menetelmiä käytetään. Ohjelmistotuotannossa on kaksi hyvää pääsääntöä:

1. Tekninen kurinalaisuus: Ohjelman tekijät tekevät ohjelman käyttäen asianmukaisia teorioita, menetelmiä ja työkaluja. Niitä pitää kuitenkin käyttää valikoivasti ja löytää ratkaisu ongelmaan vaikka sen ratkaisemiseksi ei ole suoraa teoriaa tai menetelmää. Tekijöiden pitää myös ymmärtää, että heillä on organisaation asettamat ja taloudelliset rajoitteet.
2. Kaikki ohjelmistotuotannon osa-alueet ovat yhtä tärkeitä: Ei keskitytä vain ohjelman tekniseen alueeseen, sillä ohjelmistotuotanto on paljon enemmän. Pitää ottaa huomioon projektin hallinta, kehitysvälineet ja menetelmien ja teorioiden vaikutus koko ohjelmistotuotantoon. (Sommerville 2011, 7)

Ohjelmistoprojekti voi myös olla myös osa laajempaa tuotekehitysprojektia tai laajempien tuotekehityshankkeiden osaprojekti. Ohjelman kehitysprosessissa voidaan yleensä erottaa ainakin seuraavat vaiheet: määrittely, suunnittelu, ohjelmointi ja testaus. Projektiin liittyy koko projektin ja ohjelman elinkaaren ajan kestäviä tukitoimintoja. Tärkeimmät tukitoiminnot ovat laadunvarmistus, tuotteenhallinta ja dokumentointi. Isoissa projekteissa tukitoimintoja saattaa olla enemmänkin. (Haikala 2004, 35)

Ensimmäinen julkaistu ohjelmistotuotannon vaihejakomalli oli johdettu yleisimmistä suunnitteluprosesseista. Myöhemmin vaihejakomallia ruvettiin sanomaan vesiputousmalliksi, koska jokainen työvaihe seuraa toistaan, kuin veden tippuminen vesiputouksessa. Vesiputousmallia voidaan myös sanoa ohjelman elinkaareksi, sillä se kuvaa kaikki vaiheet ohjelman teosta aina ohjelman poistumiseen käytöstä. Vesiputousmalli sisältää seuraavat vaiheet:

1. Vaatimuksien analysointi ja määrittely (Requirements definition).
2. Suunnittelu (System and software design).
3. Implementointi ja yksikkötestaus (Implementation and unit testing).
4. Integrointi ja järjestelmätestaus (Integration and system testing).
5. Ohjelman käyttäminen ja huoltaminen (Operation and maintenance). (Sommerville 2011, 31)

Jokaisessa vaihejakomallissa on hyviä ja huonoja puolia, joten jokaisen projektin alussa on valittava siihen parhaiten soveltuva vaihejakomalli. Itse valitsin vaihejakomalliksi protoilumallin, josta kerron enemmän luvussa 4. Protoilumalli sopi parhaiten ohjelman tekemiseen, sillä ohjelman vaatimuksista ei oltu varmoja. Protoilumallissa tehdään ohjelmasta prototyyppejä ja prototyyppiin lisätään tai poistetaan ominaisuuksia, niin kauan kunnes saadaan haluttu ohjelma.

3 KETTERÄ OHJELMISTOKEHITYS

Nykyään kaikki isommat ohjelmistotalot, kuten mm. Google, Yahoo, Symantec ja Microsoft käyttävät ketteriä menetelmiä. Ketterä kehitys ei ole tarkkaan määritelty prosessi, mitä voidaan seurata. Kyseessä on ennemminkin filosofia, jonka avulla voidaan ajatella ohjelmistokehitystä. Ketterän ajattelutapa on kiteytetty ketterään manifestiin, joka sisältää 4 arvoa ja 12 periaatetta. (Shore & Warden 2007, 9)

Ketterän kehityksen arvot ovat seuraavanlaiset:

1. **Yksilöt ja vuorovaikutus** ovat tärkeämpiä kuin prosessit ja työkalut.
2. **Toimiva ohjelmisto** on tärkeämpi kuin kattava dokumentaatio.
3. **Yhteistyö ja kumppanuus** ovat tärkeämpiä kuin sopimusneuvottelut.
4. **Muutoksiin reagointi** on tärkeämpää kuin suunnitelman noudattaminen. (Agilemanifesto 2001)

Vasemmalla puolella oleva lihavoitu asia määritellään tärkeämmäksi, kuin oikean puoleinen asia. Pitää muistaa kuitenkin, että oikeanpuoleiset asiat eivät ole merkityksettömiä, eikä niitä pidä unohtaa. (Ahonen 2010, 7)

Kehitysmenetelmää voidaan sanoa ketteräksi, jos se on:

1. Iteratiivinen: Sovellus kehitetään useassa kehityssyklissä.
2. Inkrementaalinen: Sovellusta ei toimiteta yhtenä valmiina pakettina, vaan sitä kehitetään ja toimitetaan pala kerrallaan.
3. Itseohjautuva: Kehittäjät määrittävät itse parhaat tavat työskennellä.
4. Yhteistoiminnallinen: Asiakas ja kehittäjät työskentelevät jatkuvassa yhteistyössä ja pitävät yllä läheistä kommunikointia.
5. Suoraviivainen: Menetelmä itsessään on helppo oppia ja se on muokattavissa sekä hyvin dokumentoitu.
6. Mukautuvainen: Viime hetken muutoksien tekeminen on mahdollista.
7. Kasvava ja kehittyvä: Prosessi, periaatteet ja työtavat kehittyvät projektin aikana sen sijaan, että ne määriteltäisiin projektin alussa. (Ahonen 2010, 7)

1990–luvulle saakka laajalle levinnyt näkemys oli, että paras tapa saada aikaan hyvä ohjelma oli tehdä huolellinen projektisuunnitelma, formalisoida laatuvarmistus ja käyttää CASE- työkalun mukaisia suunnittelu- ja analysointi metodeja, sekä täsmällisiä ohjelmistokehityksen prosesseja. Tämä ajattelutapa oli suosittu varsinkin erittäin suurien ja monimutkaisien ohjelmien tekemisessä, kuten avaruusjärjestelmien tekemisessä. Ongelmaksi tulikin kuinka saada nämä suurille ohjelmistotaloille suunnitellut toimintatavat toimimaan pienillä ja keskisuurilla ohjelmistotaloilla. Vaarana oli, että ohjelman suunnittelu vei liikaa aikaa ohjelman tekemiseltä ja suunnittelulta. Ohjelman vaatimukset myös muuttuivat, jolloin suunnittelu jouduttiin tekemään uudestaan. (Shore & Warden 2007, 58)

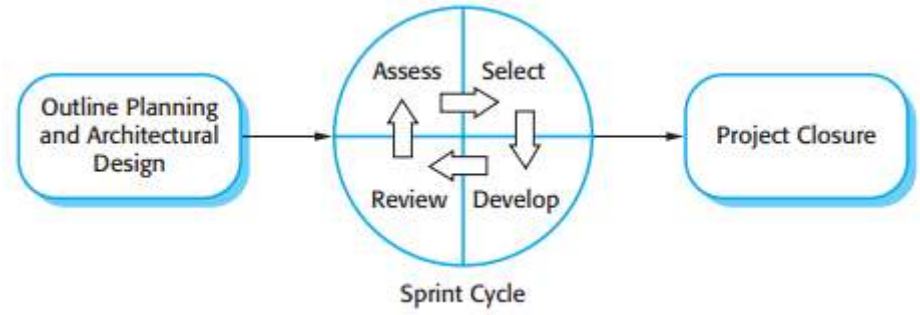
Ketterä ohjelmistokehitys antaa kehittäjille mahdollisuuden keskittyä paremmin itse ohjelmaan, sillä se luottaa inkrementaaliseen eli vähittäin kasvavaan lähestymistapaan ohjelman määrittelyssä, kehittelyssä ja toimituksessa. Ketterä ohjelmistokehitys toimii parhaiten projekteissa, joissa vaatimukset muuttuvat nopeasti ohjelman tekemisen aikana. Tarkoituksena on tehdä ohjelman tilaajalle mahdollisimman nopeasti valmis ohjelma ja vähentää ohjelman tekemisessä olevaa byrokratiaa ja turhien dokumenttien syntymistä. Valmiista ohjelmasta tilaaja antaa palautetta ja ohjelmaan lisätään tai muokataan ominaisuuksia kunnes ohjelma on kokonaan valmis. (Shore & Warden 2007, 58-59).

Koska ketterät menetelmät toimivat parhaiten pienissä ja tiukasti integroiduissa ryhmissä, ongelmaksi muodostuu ryhmän skaalautuvuus suuriin järjestelmiin. Ongelma tulee vastaan varsinkin tehtäessä kriittisiä ohjelmia, kuten ydinvoimalan tai junaradan ohjausjärjestelmiä. Syynä ovat järjestelmien vaatimat erittäin tiukat tietoturva- ja luotettavuusanalyysit. Tämän takia ketteriin menetelmiin pitäisi tehdä erittäin paljon muutoksia, ennen kuin niitä voi käyttää kriittisten ohjelmien tekemiseen. (Shore & Warden 2007, 60).

3.1 Scrum

Scrum-menetelmän kehitti Jeff Sutherland 1990-luvun alkupuolella ja sitä on tämän jälkeen kehitetty eteenpäin. Scrum-menetelmän painopiste on iteratiivisessa kehittämisessä. Iteratiivinen kehittäminen tarkoittaa saman asian tekemistä useaan kertaan. Scrum ei määrittele käytettäviä ohjelmoinnin käytäntöjä, kuten pariohjelmointia, vaan siihen voidaan ottaa muita ketteriä menetelmiä mukaan, kuten Extreme Programming. (Sommerville 2011, 72)

Kuvasta 1 voidaan huomata Scrum-menetelmän jakautuvan kolmeen vaiheeseen. Ensimmäiseksi tulee suunnittelu ja luonnosvaihe, missä vahvistetaan yleiset tavoitteet hankkeelle ja suunnitellaan ohjelmistoarkkitehtuuri. Tämän jälkeen tulevat sprintit, jotka ovat iteratiivisia kierroksia ja jokaisessa syklissä lisätään ominaisuuksia ohjelmaan. Lopuksi on projektin päättäminen eli tässä vaiheessa ohjelman vaatimukset on toteutettu, eikä ohjelmaan tarvitse lisätä ominaisuuksia. (Sommerville 2011, 73)



Kuva 1. Scrum-prosessi (Sommerville 2011, 73)

Sprint kierto on Scum-menetelmän innovatiivisin ja keskeisin osa. Sprintit kestävät yleensä 2-4 viikkoa. Jokaisen sprintin alussa pidettävässä palaverissa asiakas tai muu tuotteen tilaaja määrittelee tärkeysjärjestyksen työlistassa oleville töille. Tämän jälkeen Scrum-kehitysryhmä valitsee tulevan sprintin aikana tehtävissä olevat työt ja työt toteutetaan. Lopuksi ohjelma esitellään ohjelman tilaajalle ja ohjelma arvioidaan. Arvioinnin jälkeen aloitetaan sprintti uudestaan. Sprinttejä toistetaan niin kauan, että ohjelmassa on kaikki tarvittavat ominaisuudet ja toiminnot. (Sommerville 2011, 73)

Scrum-menetelmän perusidea on antaa koko tiimille mahdollisuus tehdä valintoja, jotka ovat olleet yleensä projektipäällikön vastuulla. Yksi henkilö ryhmästä toimii Scrum-mestarina (Scrum Master), jonka tehtävänä on järjestää päivittäiset tapaamiset, ohjelmaan saatujen ominaisuuksien listan päivittäminen ja kommunikoiminen tiimin ja tilaajan kanssa. Päivittäisen tapaamisen tarkoituksena on tiedon jakaminen. Tiedon jakaminen takaa, että kaikki tiimin jäsenet tietävät missä mennään projektissa, sekä mitä ominaisuuksia ohjelmaan tarvitsee tehdä. Tapaamiset pidetään yleensä lyhyinä, jotta ei kuluteta turhaan aikaa. (Sommerville 2011, 73)

3.2 Scrum ja protoilumalli

Protoilumallissa ja scrumissa on iteratiivisia toimintoja, joita tehdään niin kauan kunnes ohjelma saadaan valmiiksi. Scrumissa ohjelma rakentuu pikkuhiljaa valmiimmaksi ja täydemmäksi. Ohjelma annetaan myös tilaajan testattavaksi. Prototyyppimallissa tehdään prototyyppejä, jotka eivät ole vielä valmiita ja niiden avulla pyritään saamaan palautetta ohjelman tilaajalta. Prototyyppien on tarkoitus auttaa ainoastaan ohjelman teossa, sillä vasta prototyyppien jälkeen ohjelmaa aletaan kehittää valmiiksi järjestelmäksi. Ohjelman tilaaja antaa palautetta prototyypeistä ja palautteen pohjalta tehdään tarvittavat muutokset.

Scrumissa projektipäällikkö on Scrum-mestari ja hän huolehtii siitä, että tiimi voi tehdä työtä optimaalisesti ja että scrumia noudatetaan oikein. Scrum-mestari, kuten myös projektipäällikkö, kommunikoi eri sidosryhmien kanssa ja välittää heiltä saamansa tiedot ohjelman tekijöille.

Yleisesti ketterissä menetelmissä suositaan suoraa viestintää, mielellään kasvokkain, kun taas prototyyppisessä mallissa keskitytään enemmän dokumenttien tekemiseen. Ketterät menetelmät myös korostavat toimivan ohjelmiston olevan ensisijainen edistyksen mittari, toisin kuin protoilumallissa. Ketterissä tehdään suunnittelua jatkuvasti, koko projektin keston ajan.

4 PROTOILUMALLI

Protoilumallissa ohjelmasta tehdään prototyyppi, jossa testataan ohjelmassa olevia ominaisuuksia ja käyttöön tulevaa tekniikkaa. Erityisen hyödylliseksi protoilu onkin osoittautunut juuri käyttöliittymien määrittelyn yhteydessä, sillä silloin voidaan tehdä erilaisia versioita käyttöliittymästä. Myös erilaiset suorituskykyä testaavat prototyypit ovat yleisiä, sillä silloin voidaan testata erilaisia teknisiä ratkaisuja (Haikala 2004, 43).

Prototyypit soveltuvat erityisesti uusien teknisten vaatimuksien kokeilun tekemiseen tai etsittäessä epäselviä asiakasvaatimuksia. Valmistuneen prototyypin kaksi pääkäyttövaihtoehtoa ovat seuraavat:

1. Prototyypin valmistuttua sen perusteella määritellään toteutettava järjestelmä, joka sitten toteutetaan alusta alkaen uudelleen.
2. Prototyyppi kehitetään valmiiksi järjestelmäksi. Luonnollisesti myös välimuodot ovat mahdollisia. (Haikala 2004, 42)

Protoilumallissa on kaksi erilaista päävaihtoehtoa, evoluutioprototyyppi (evolutionary prototype) ja kertakäyttöinen prototyyppi (throw-away prototype). Evoluutioprototyyppiä kehitetään vaiheittain valmiiksi ohjelmaksi, ja kertakäyttöprototyyppillä pystytään mallintamaan järjestelmää, minkä jälkeen varsinaisen tuotteen kehittäminen aloitetaan alusta. On myös mahdollista toteuttaa ohjelma käyttäen molempien lähestymistapojen välimuotoja, jolloin joitakin prototyyppejä pyritään hyödyntämään, mutta jotkut osat valmistaudutaan korvaamaan uudella toteutuksella. (Haikala 2011, 38)

Evoluutioprototyypin suurin ongelma on, että huonosti toimiva prototyyppi jää helposti osaksi lopullista järjestelmää. Vaarana on, että jonkin prototyypin komponentti saattaa toimia lähes oikean komponentin tapaan, mutta olla toteutukseltaan tavattoman hidas. Tällaisen toteutuksen korvaaminen toimivalla nopeammalla komponentilla saattaa unohtua tai pahimmassa tapauksessa asia saattaa kokonaan jäädä huomaamatta testausvaiheessa, jolloin totuus paljastuu vasta ohjelman valmistuttua. (Haikala 2011, 39)

Yksi esimerkki kertakäyttöisestä prototyypistä on käyttöliittymän suunnittelu. Käyttöliittymä voidaan esitellä ohjelman tilaajalle yksinkertaisimmillaan piirtämällä paperille joko näytön tapahtumat tai pelkkä navigointi. Navigointia esitellessä voidaan myös näyttää, miten siirtyminen näytöltä toiselle tapahtuu. Tämä on erittäin hyvä tapa näyttää tilaajalle, minkälaista ohjelmaa ollaan tekemässä. (Haikala 2011, 39)

Ehkä suurin ongelma protoilussa tulee vastaan silloin, kun esittelee asiakkaalle prototyyppiä ja asiakas luulee, että kyseessä on valmis ohjelma, vaikka käytännössä valtaosa työstä on vielä tekemättä. Tästä syystä ei kannata tehdä mahdollisimman viimeistellyn näköistä ja tuntuista prototyyppiä, koska asiakkaan on huomattava myös, että järjestelmä on vielä keskeneräinen. (Haikala 2004, 43)

Protoilumalli vie myös runsaasti aikaa, koska voidaan joutua tekemään paljon erilaisia ratkaisuja. Varsinkin nettisovelluksissa ulkoasun prototyyppejä voidaan joutua tekemään useita, mikäli ei tiedetä millainen ulkoasu halutaan ohjelmaan. Tämän takia protoilumalli ei välttämättä ole oikea, jos projekti pitää saada valmiiksi tiukassa aikataulussa.

5 PROTOILUMALLIN VAIHEET

5.1 Esikartoitus ja määrittely

Esikartoitusta voidaan myös kutsua esitutkimukseksi ja sen pitäisi vastata kysymyksiin:

1. Mitä järjestelmän pitäisi tehdä?
2. Onko järjestelmää mahdollista tehdä?
3. Onko järjestelmää kannattavaa tehdä?

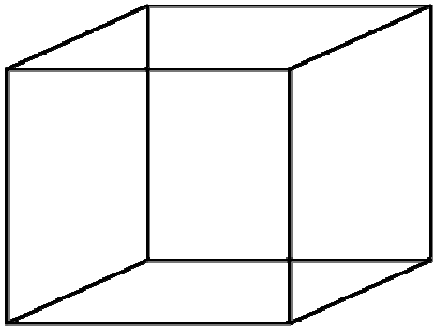
Esikartoituksen tarkoitus onkin selvittää kannattaako projektin suunnittelua jatkaa vai peruutetaanko se kokonaan. Tämä on tärkeää tehdä ensimmäiseksi, ettei käytetä resursseja turhaan. (Haikala 2004, 92)

Avainasemassa esikartoituksessa ovat asiakasvaatimukset. Asiakasvaatimuksien pitää olla oikein ymmärrettyjä ja pysyä mahdollisimman muuttumattomina. Vaatimusmuutokset vaikuttavat kaikkiin ohjelmistotyön vaiheisiin, joten mitä myöhäisemmässä vaiheessa niitä tehdään, sitä enemmän lisätyötä ne aiheuttavat ja voivat pahimmassa tapauksessa siirtää ohjelman valmistumisajankohtaa myöhemmäksi. (Haikala 2004, 94)

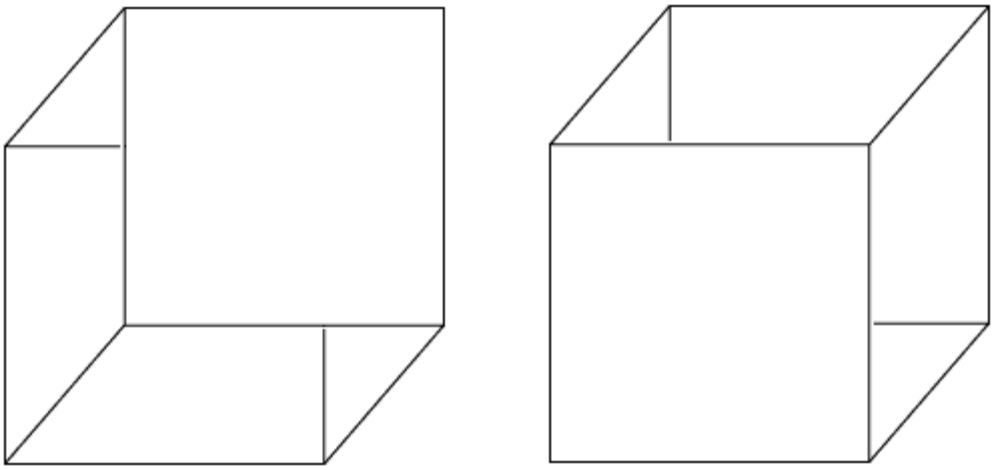
Asiakasvaatimuksia tehtäessä ja niitä miettiessä kannattaa olla mahdollisimman monta ihmistä mukana. Asiakas ei välttämättä tiedä kaikkia ohjelman vaatimuksia kovinkaan tarkkaa ja kannattaako niitä edes tehdä valmiiseen ohjelmaan. Tärkeää on, ettei ole liian kriittinen ideoille, vaan niitä kannattaa heitellä ilmaan. Aluksi huonolta kuulostava idea voi olla lopulta loistava. On olemassa paljon erilaisia tapoja kerätä ohjelman vaatimuksia, kuten keskustelut ja niiden kirjoittaminen muistilapuille ja tämän jälkeen mietitään ryhmässä mitkä vaatimukset ovat hyviä. Mikäli yksi tapa ei tuo tuloksia, kannattaa kokeilla toista. (Pilone & Miles, 34)

Alustavat asiakasvaatimukset ovat lähestulkoon aina puutteellisia ja toisaalta ne ovat usein myös keskenään ristiriitaisia. Esikartoitusta pitääkin päivittää hyvin usein ja tämä tekee suurista tuotekehityksistä vaikeita, sillä niissä on valtavasti vaatimuksia. Ohjelmiston käyttäjät voivat vaatia hyvinkin erilaisia asioita tai he voivat olla ristiriitaisia muiden vaatimuksien kanssa. Asiakasvaatimusten kerääminen tuotteen seuraavia versioita varten on koko tuotteen elinkaaren kestävä toiminto. (Haikala 2004, 92)

Vaatimusten muuttumista voidaan esittää graafisesti esim. Neckerin kuutiolla (kuva 2). Kyseessä on kuutio joka vaihtaa ”suuntaa”, kun sitä on katsonut tarpeeksi kauan. Ensimmäinen näkemäsi ”suunta” on ensimmäinen asiakkaan kanssa tehty vaatimus ohjelmistosta ja seuraava ”suunta” lopullinen ohjelma (kuva 3). Tällöin voidaan huomata, että ohjelma saattaa olla hyvinkin erilainen kuin aluksi luultiin. (Kosko 1993)



Kuva 2. Neckerin kuutio (BBC 2012)



Kuva 3. Neckerin kuutiossa näkyvät suunnat (BBC 2012)

Määrittelyvaiheessa kootaan kaikki asiakkaan kanssa sovitut vaatimukset ja analysoidaan ne. Tämän jälkeen vaatimuksista tehdään ohjelmistovaatimukset, joiden pohjalta toteutetaan ohjelma. Tämä voi olla joissakin tapauksissa vaativaa, mikäli esikartoitusta ei ole tehty hyvin. Määrittelyn tuloksena syntynyttä dokumenttia sanotaan toiminnalliseksi määrittelyksi. (Haikala 2004, 39)

Toiminnallisessa määrittelyssä kerrotaan ohjelmiston toiminnot, toteutukselle asetettavat ei-toiminnalliset vaatimukset sekä rajoitukset. Toimintojen yhteydessä määritellään ohjelmistolla toteutettavat ominaisuudet, käyttöliittymä ja kommunikointi muiden järjestelmien kanssa. Ei-toiminnallisia vaatimuksia ovat esim. suoritusteho, vasteaika ja käytettävyys. Rajoituksia tulee tietysti käytettävästä järjestelmästä, levytilasta ja ohjelmointikielistä jne. Määrittelyvaiheessa on siis kyse asiakasvaatimusten muuntamisesta täsmällisiksi ohjelmistovaatimuksiksi. (Haikala 2004, 39)

5.2 Arkkitehtuurisuunnitelma ja moduulisuunnittelu

Kun on saatu asiakkaan vaatimukset täsmälliseksi ohjelmistovaatimukseksi, tehdään arkkitehtuurisuunnitelma. Arkkitehtuurisuunnitelmassa syntyy järjestelmästä koko järjestelmän kattava kuvaus, jota sanotaan tekniseksi määrittelyksi. (Haikala 2004, 43)

Arkkitehtuurisuunnitelman tarkoituksena on siis muuntaa asiakkaan tarpeiden mukaan tehty määrittely tekniselle kielelle. Arkkitehtuurisuunnittelua seuraa moduulisuunnitteluvaihe, jossa jokaisen moduulin sisäinen rakenne suunnitellaan. Termillä moduuli on monia eri tulkintoja, mutta tässä tekstissä moduuli on ohjelmasta erotettavissa oleva looginen kokonaisuus, kooltaan tyypillisesti alle 1000 ohjelmariviä. Tyypillinen moduuli sisältää tietomäärittelyitä ja joukon ko. tietoa käsitteleviä funktioita. (Haikala 2004, 40)

Arkkitehtuurisuunnittelu on suurimmaksi osaksi osien välisen työnjaon ja rajapintojen suunnittelua. Tavoitteena on pyrkimys mahdollisimman vähän toisistaan riippuviin moduuleihin niin, että yksityisen moduulin sisällä tehtävät muutokset eivät vaikuta moduulin ulkopuolelle. (Haikala 2004, 82)

5.3 Prototyyppi ja tarkistus

Opinnäytetyössä järjestelmästä tai järjestelmän osasta tehdään prototyyppi ja se annetaan tarkastettavaksi tietyn väliajoin asiakkaalle. Tarkastuksen jälkeen tehdään tarvittavat muutokset ja tätä toistetaan niin kauan, että saadaan sovellus missä kaikki halutut ominaisuudet on ohjelmoitu.

Tarkastuksessa saattaa myös tulla ilmi uusia vaatimuksia ja tällöin pitää miettiä, kannattaako ne tehdä prototyyppiin vai tehdäänkö ne vasta lopulliseen ohjelmaan. Vaatimuksien lisäämisessä kannattaa muistaa, että ominaisuuksia ei kannata lisätä ohjelman prototyyppiin loputtomasta, vaan kannattaa tehdä prototyyppi valmiiksi ohjelmaksi ja vasta tämän jälkeen lisätä ominaisuuksia. Tällöin voidaan säästää huomattavasti aikaa.

Prototyyppivaiheessa kannattaa myös tehdä ohjelman ulkoasusta prototyyppi, sillä tällöin voidaan selvittää siinä olevat virheet ja ongelmat. Varsinkin web-pohjaisissa järjestelmissä www-sivujen ulkoasun ongelmat kannattaa tarkistuttaa, sillä ulkoasu riippuu myös käytetystä selaimesta.

5.4 Asennettavan järjestelmän teko ja käyttöohje

Asennettava järjestelmä tehdään arkkitehtuurin mukaisesti kokonaan valmiiksi. Prototyypin koodi voi olla hyvin vaikeaselkoista ja kommentoimatonta, mutta asennettavassa järjestelmässä sen on oltava hyvin kommentoitua ja edettävä suunnitellusti, sillä myöhemmin valmista ohjelmakoodia voidaan joutua muokkaamaan. Kunnollisesti kommentoitua koodia on helppo myöhemmin muokata, eikä aikaa kulu koodin ymmärtämiseen.

Käyttöohjeen tekeminen voi vaikuttaa ensi silmäykseltä helpolta tehtävältä, mutta kyse on varsin vaativasta tehtävästä. Käyttöohjeiden tehtävä on kertoa ohjelman käyttäjälle ohjelman käytöstä tekstin ja kuvien avulla mahdollisimman yksinkertaisesti. Ennen käyttöohjeen kirjoittamista pitää päättää minkä tasoiselle tietokoneen käyttäjälle kirjoitetaan ohjetta, tämä vaikuttaa siihen minkälaista sanastoa voidaan käyttää ohjeessa.

Käyttöohjeen suunnittelussa pitää ottaa huomioon se, että kaikilla ohjelman käyttäjillä ei ole suurta kokemusta tietotekniikasta. Ongelmia aiheuttavat vaikeat tietotekniset alan sanat, koska ne pitäisi pystyä kertomaan siten, että kaikki ymmärtävät ne. Ohjeiden tekemisessä kannattaakin käyttää kuvia, sillä ne ilmaisevat asiat paljon helpommin kuin teksti. (Barnett 2008, 20)

Ennen käyttöohjeen kirjoittamista, kannattaa miettiä vastaus seuraaviin kysymyksiin:

1. Minkälaiset ihmiset tulevat käyttämään ohjetta?
2. Mitä jokaisen käyttäjän pitää löytää ohjeista?
3. Mitä hyötyjä eri lukijat saavat ohjeista?
4. Kuinka paljon yksityiskohtia tarvitaan?
5. Jos annamme ohjeen käyttäjälle, lukeeko hän sen?
6. Missä järjestyksessä asiat esitetään?
7. Minkälaisia teknistä sanastoa käytetään?
8. Pitääkö ohjeisiin laittaa sanasto selittämään teknisiä sanoja?

Ohjeiden pitäisi edetä loogisesti eteenpäin. Esim. mikäli kyseessä on web-sovellus, kannattaa eri sivujen toiminnot kertoa erikseen ja aloittaa etusivulta, mikäli web-sovelluksessa on sellainen. Web-sovelluksissa voi tulla ongelmaksi käyttöliittymä, sillä www-sivujen käyttöliittymät ovat hyvin erilaisia.

5.5 Testaus

On suositeltavaa, että ohjelman testaa jokin muu kuin ohjelman tekijä. Ulkopuolisella testaajalla on parempi mahdollisuus löytää enemmän virheitä, sillä hän ei tiedä kunnolla, miten ohjelma toimii ja hän voi mm. löytää ihan uusia ongelmia järjestelmästä.

Testauksen tavoitteena on etsiä mahdolliset virheet ja häiriöt ohjelmasta. Tämä on yleensä hankalaa, sillä suuret ohjelmistot ovat vaikeita testattavia ja testaaminen ei välttämättä kerro, toimiiko ohjelma oikein kaikissa mahdollisissa tilanteissa. Verkkosovelluksissa ongelmia tuottavat erilaiset selaimet, sillä joissakin tapauksissa ne voivat lähettää virheellistä dataa palvelimelle. Selaimesta on mahdollisuus myös ottaa pois käytöstä esim. JavaScript, jolloin sovellukset jotka tarvitsevat sitä eivät toimi ollenkaan.

Verkkosovelluksen testaamisessa testataan myös palvelimet ja tietokannat. Tämä on hieman hankalampaa kuin ns. työpöytäsovelluksen, koska verkkosovellus vaatii nettiyhteyden ja on riippuvainen enemmän palvelimesta ja selaimesta kuin käyttäjän omasta tietokoneesta. Verkkosovellusten ylläpito on myös suuressa roolissa testauksessa, sillä useimpien sovellusten sisältö päivittyy säännöllisesti. Tätä varten on oltava olemassa työkalut mahdollisimman helppoon ylläpitoon. (Mäki 2008)

Yksinkertaisen testaus voidaan toteuttaa koodausvaiheessa ohjelmointikielten kääntäjien avulla. Kääntäjissä olevilla automatisoiduilla testityökaluilla voidaan testata vain tiettyyn pisteeseen saakka ja testaaminen on hyvin rajoittunutta. Testityökaluilla pystytään testaamaan virheiden lisäksi web-palvelimien suorituskykyä erilaisissa kuormituksissa. Kuormansietojen testaaminen ja sen anta-

mien tuloksien tulkitseminen oikein voi olla vaikeaa. Joissakin tapauksissa ongelma saattaa olla ohjelman sijaan verkkolaitteissa tai tietokannassa.

Myös tietoturvan testaaminen on lisätty yleisimpiin kehitystyökaluihin, kuten esim. Microsoftin Visual Studioon. Kyseinen ohjelma tunnistaa yleisimmät tietoturvariskit ja yrittää selvittää muistin ylivuotojen riskejä, verkkoyhteyksien käyttöä ja käyttöjärjestelmän tietoturvakäytäntöjen noudattamista. (Lehto 2005)

5.6 Ohjelman toimitus ja käyttöönotto

Web-sovellus asennetaan asiakkaan palvelimille ja ohjelmaan, tietokantaan ja palvelimeen tehdään tarvittavat muutokset, jotta ohjelma toimisi oikein. Samalla pitää tarkistaa asiakkaan palvelimen palvelinohjelmistot, sillä ne saattavat olla vanhentuneet. Vanhentuneissa palvelinohjelmistoissa on yleensä tietoturvaongelmia. Ohjelman asennuksen aikana voidaan myös tehdä palvelimelle stressitesti, jolloin voidaan arvioida ohjelman käyttäjämäärän vaikutus ohjelman käyttöön.

Ennen käyttöönottovaihetta pidetään käyttäjille koulutus. Koulutuksen tarkoituksena on kertoa ohjelman toiminnoista ja kuinka ohjelmaa voidaan käyttää tehokkaimmin. Ohjeiden olisi hyvä olla saatavilla tekstimuodossa, koska se on yleensä tutuin tapa tarkistaa asioita. Ohje voidaan myös lisätä itse ohjelmaan, jolloin ohjeiden lukeminen hoituu nopeasti.

Käyttöönotossa otetaan ohjelma jokapäiväiseen käyttöön. Ohjelman käyttöönoton jälkeen siinä voi olla ongelmia, kuten ohjelmointivirheitä. Ohjelmointivirheet pitää korjata mahdollisimman nopeasti. Pahimmillaan ohjelmointivirheen korjaamisen ajaksi ohjelman käyttö pitää lopettaa. Edellä mainittua ongelmaa voidaan välttää tekemällä ohjelman testaaminen hyvin.

5.7 Ylläpito

Ohjelman ylläpitäminen on ohjelman suuruudesta riippuen hyvinkin aikaa vievää, sillä siihen kuuluu kaikki käyttöönoton jälkeen tapahtuvat muutokset ja korjaukset. Verkkosovellusten ja tietokantojen ylläpitäminen lisää ohjelman ylläpitäjien työtaakkaa, sillä varsinkin tietokannat ja serverit ovat hankalia korjata nopeasti. Pahimmillaan palvelu voi olla kokonaan poissa käytöstä viikkoja.

Ylläpitoon kuuluu myös uusien tietoturvariskien korjaaminen, sillä niitä voi ilmetä ohjelman käytön yhteydessä. Varsinkin vanhemmissa ohjelmissa voi olla vakavia tietoturvariskejä, jolloin pahimmassa tapauksessa asiakkaan tietoja voi päättyä vääriin käsiin. Tästä syystä pitää yrityksen olla hyvin varuillaan uusista tietoturvaongelmista.

6 TYÖSSÄ KÄYTETYT TEKNIIKAT

Kyseessä on verkossa toimiva ohjelma, joten ohjelmointikielinä toimivat PHP, HTML, JavaScript ja CSS. Nämä tekniikat mahdollistavat sen, että ohjelmasta tulee monipuolinen ja helppokäyttöinen. Ohjelman tekemisessä olisi voinut käyttää enemmänkin valmiita sovelluksia, kuten Googlen kalenteria. Googlen kalenteri olisi sopinut hyvin aikojen varaukseen, jolloin ei olisi tarvinnut itse tehdä kalenteria, mutta päätin tehdä sen itse, koska en ollut ennen käyttänyt kyseistä komponenttia ja en olisi tarvinnut kaikkia Google kalenterin tarjoamia ominaisuuksia.

6.1 PHP

Vuonna 1995 Rasmus Lerdorf halusi tietää kuinka moni käyttäjä kävi lukemassa hänen ansioluetteloaan. Tuolloin ei ollut vielä valmiita työkaluja kyseiselle toiminnolle, joten hänen piti itse ohjelmoida ne käyttäen Perliä ja CGI:tä. Skriptistä tuli suosittu ja moni pyysi sitä sähköpostin kautta, samalla hän antoi työkalulleen nimeksi Personal Home Page (PHP). Vuonna 1997 hän julkaisi PHP:stä version 2.0 ja siirtyi Perlistä käyttämään C:tä. Tarkemmin versiota 2.0 kutsuttiin nimellä Personal Home Page/Form Interpreter (PHP/FI), koska se mahdollisti HTML-lomakkeiden tietojen käsittelemisen. PHP 2.0 -julkaisun jälkeen PHP:stä tuli erittäin suosittu ja tämän jälkeen sitä on kehitetty eteenpäin ja nykyään se tukee olio-ohjelmointia ja sisältää sisäänrakennetun tietokantamoottorin (SQLite). Tällä hetkellä uusin versio on 5.3.9. (Gilmore 2008, 2)

Itse PHP-koodi käännetään vasta palvelimella, joten palvelimelle pitää olla ensin asennettu PHP-tuki, jotta PHP-tiedostot toimivat. Koodi voidaankin kirjoittaa millä tahansa tekstinkäsittelyohjelmalla ja siirtää tämän jälkeen palvelimelle. PHP-koodi tallennetaan .php-tiedostopäätteellä. PHP mahdollistaa myös HTML-koodin kirjoittamisen PHP-koodin sekaan. (Boronczyk ym. 2009, 20)

PHP-koodin hyvä puoli on myös, että ei ole väliä miten koodi on jäsennelty tai rivitetty, koska palvelimelle koodi näkyy yhtenä pitkänä rivinä. Tämä antaa ohjelmoijalle vapauden muotoilla koodin siten kuin hän parhaimmaksi näkee, mut-

ta kannattaa pitää mielessä, että joku muukin voi joutua lukemaan tai muokkaamaan koodia. Siksi kannattaakin miettiä minkälaisia sisennyksiä käyttää, sillä koodin pitää olla helppolukuista. (Boronczyk ym. 2009, 21)

PHP-ohjelmointikieli sisältää joitakin ominaisuuksia, joiden runsas käyttäminen voi hidastaa ohjelman suorittamista. Olen listannut joitakin nopeuttavia tekijöitä, joita kannattaa ottaa huomioon ohjelmia tehtäessä:

1. Echo on nopeampi kuin print.
2. Käytä muuttujien tulostamisessa (echo) pisteen sijaan pilkkua. Esim. (echo "Testi: ",$muuttuja). Pilkut ovat hieman nopeampia, kuin pisteet.
3. Käytä require()- ja include() -funktioita require_once()- ja include_once() -funktioiden sijaan.
4. Pelkkä PHP -kielen käyttäminen staattisen HTML -koodin sijaan voi olla 2-10 kertaa hitaampaa, kun käytössä on Apache -palvelin. Käytä enemmän staattista HTML-koodia ja itse ohjelman tapahtumiin PHP-koodia. (Wars 2009)

6.2 CSS ja ulkoasun suunnittelu

CSS (Cascading Style Sheets) mahdollistaa erilaisien ulkoasuun liittyvien sääntöjen antamisen www-dokumenteille. CSS:llä voidaan hyvin monipuolisesti vaikuttaa ulkoasuun, niin ulkoasun asetteluun kuin käytettävissä oleviin väreihin. CSS:ää käytettäessä on pidettävä mielessä, että selaimet lukevat hieman eri tavalla sen standardeja. Ulkoasu pitääkin testata eri selaimilla, jotta tiedetään varmasti sivujen toimivuus.

Itse testasin ulkoasua Operalla, FireFoxilla, Chromella ja Internet Explorerilla. Sivut toimivat hyvin kaikilla muilla selaimilla paitsi Internet Explorerilla. Explorerilla oli vaikeuksia lukea eri elementtien korkeudet oikein ja tällöin sivut eivät olleet halutun näköiset. Onneksi kyseessä oli muutaman pikselin heitto ja sen korjaaminen ei ollut vaikeaa, otettiin vain korkeudesta muutama pikseli pois.

Aina kun lähdetään miettimään ohjelman ulkoasua, pitää ottaa huomioon myös käytettävyys. Ulkoasua suunnittelussa pitää ottaa huomioon käytettävyys - ei kukaan halua käyttää ohjelmaa jonka käytettävyys on huono. Jakob Nielsen, joka on ehkä tunnetuin Webin käytettävyyden tutkija, on kiteyttänyt käytettävyy-

den hyvin: "Webissä asiakas on kuningas, hiiri aseenaan hän päättää kaikesta. Jos palvelu ei tyydytä, asiakkaan on helppo mennä muualle, koska myös kaikki kilpailijat ovat vain hiiren liikautuksen päässä". (Nielsen 2000, 9)

Ulkoasun suunnittelussa pitää ottaa huomioon myös hitaat Internet-yhteydet ja tietokoneet. Jakob Nielsen on tullut tutkimuksissaan siihen tulokseen, että käyttäjä haluaa sivujen latautuvan nopeasti. Sivulta toiselle siirtyessä vasteajan pitäisi olla alle sekunti, jotta käyttäjä tuntee liikkuvansa vapaasti. Esimerkiksi IBM teki 70- ja 80-luvuilla tutkimuksen josta selvisi, että keskuskoneiden käyttäjät toimivat tuottavimmin, kun aika näppäimen painamisesta halutun ruudun esiin saamiseksi oli alle yksi sekunti. (Nielsen 2000, 42)

Käytettävyyteen voidaan myös lukea sivujen kirjallinen tyyli, sillä Internet-sivuille pitää kirjoittaa lyhyesti ja ytimekkäästi. Internetissä tekstin silmäiltävyys on hyvin tärkeää. Seuraavat kolme sääntöä ovat pääosassa Webiin kirjoittaessa:

1. Kirjoita ytimekkäästi: Käytä tekstin tuottamiseen vain puolet siitä sanamäärästä, jonka olisit käyttänyt saman materiaalin käsittelyyn paperilla.
2. Kirjoita teksti silmäiltäväksi. Käyttäjiä ei pidä pakottaa lukemaan pitkiä tekstijaksoja; teksti on jaoteltava osiin lyhyiden kappaleiden, alaotsikoiden ja luetteloiden avulla.
3. Jaa pitkä teksti hypertekstin keinojen avulla useammille sivuille. (Nielsen 2000, 101)

Kuinka pitkän ajan käyttäjät oleskelevat Internet-sivuilla?

Vuonna 2010 Microsoftin tutkimuskeskus julkaisi artikkelin, jossa tarjotaan matemaattista tietoa ja mallia web-sivuilla käytetyn ajan ja sivuilta poistumisen välisestä suhteesta. Tutkimuksen kohteeksi valittiin Internet-sivut, joiden käyntimäärä oli yli 10 000 ja sivuja oli yhteensä 205 873 kappaletta. Tutkijat huomasivat käyttäjien sivuilla kuluttaman ajan ja Weibull-jakauman (Weibull distribution) kanssa yhtäläisyyksiä. (Nielsen 2011)

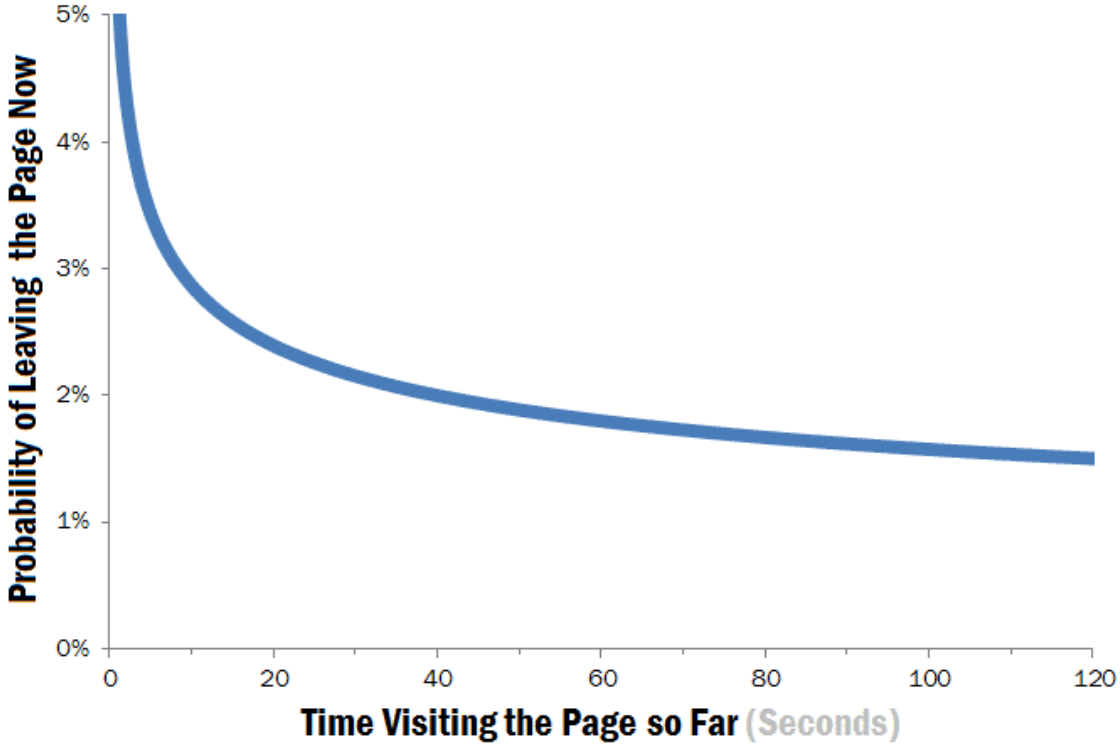
Weibull-jakauma kuuluu todennäköisyysteoriaan ja sitä käytetään luotettavuuden suunnittelussa. Lyhykäisyydessään sillä analysoidaan komponentin hajoamiseen kuluva aika. Tällöin voidaan tehdä arvio milloin osa pitää vaihtaa uuteen. Tietysti kun arvioidaan sivujen vierailuja pitää "osan hajoaminen" korvata sanalla "käyttäjä lähtee sivuilta". Tutkimuksessa todistettiin, että sivuilla käytetty aika voitiin laskea suurin piirtein käyttämällä Weibull-jakaumaa. (Nielsen 2011)

Weibull-jakaumaa käyttämällä voidaan laskea osan ikääntymiselle kaksi erilaista todennäköisyyttä:

1. Positiivinen ikääntyminen: Mitä pidempään komponentti on ollut käytössä, sitä todennäköisempää on sen hajoaminen.
2. Negatiivinen ikääntyminen: Mitä pidempään komponentti on käytössä, sitä epätodennäköisempää on sen hajoaminen. Kyseessä on erittäin hyvälaatuinen osa, joka kestää pidempään. Huonolaatuiset osat hajoavat nopeasti. (Nielsen 2011)

Tutkimuksessa 99 % valituista sivuista kuului negatiivisen ikääntymisen jakaumaan. Näin vahva löytö on erittäin harvinaista ihmisen ja tietokoneen vuorovaikutusta (HCI, human-computer interaction) tutkivassa tutkimuksessa. Negatiiviseen ikääntymiseen päädyttiin, koska web-sivujen laatu vaihtelee erittäin paljon. Käyttäjät tietävät tämän ja lähtevät sivuilta mahdollisimman nopeasti. On harvinaista, että käyttäjät viipyvät Internet-sivuilla pitkään, mikäli kuitenkin käyttäjä löytää sivuilta mielenkiintoista lukemista tai tekemistä, hän voi oleskella niillä pidempään. (Nielsen 2011)

Kuvio 1 kertoo todennäköisyyden lähteä sivuilta suhteutettuna sivuilla käytettyyn aikaan.



Kuvio 1. Käyttäjän lähteminen sivuilta (Nielsen 2011)

Kuviosta 1 voidaan huomata, että ensimmäiset 10 sekuntia ovat kriittiset hetket jääkö vai lähteekö käyttäjä sivuilta. Korkea todennäköisyys käyttäjän lähtemiseen sivuilta ensimmäisien sekuntien aikana johtuu siitä, että käyttäjät ovat erittäin epäileviä. Epäileväisyys johtuu siitä, että he ovat joutuneet käyttämään lukuisia huonosti suunniteltuja web-sivuja. Ihmiset tietävät, että useimmat websivut ovat hyödyttömiä ja he haluavat kuluttaa mahdollisimman vähän aikaa niiden selaamiseen. (Nielsen 2011)

Jos sivuilla vietetään yli 10 sekuntia, käyttäjät tutkivat sivuja hieman tarkemmin. Kuitenkin on todennäköistä, että he lähtevät seuraavan 20 sekunnin aikana. Kuvaajan mukaan noin 30 sekunnin kohdalla voidaan huomata, että käyttäjät lopettava sivujen käytön paljon pienemmällä tahdilla. Joten jos pystyy pitämään käyttäjän puoli minuuttia, on paljon parempi mahdollisuus, että käyttäjä pysyy pidempään. Pitää kuitenkin muistaa, että netissä puoli minuuttia on pitkä aika.

Karkeasti sanottuna tuloksesta voidaan tehdä kaksi päätelmää:

1. Huonot sivut: Sivuilla vieraillaan muutama sekunti.
2. Hyvät sivut: Sivuilla vieraillaan muutama minuutti.

Käyttäjän tekemä päätös sivujen hyvyydestä ja huonoudesta tehdään ensimmäisien sekuntien aikana. Sivujen suunnittelun tärkeys tuleekin erittäin tärkeäksi ensimmäisien sekuntien aikana. Saadaksesi käyttäjä käyttämään muutaman minuutin sivuilla pitää se näkyä heti ensimmäisen 10 sekunnin aikana. Tällöin hyvin suunnitellut sivut erottuvat edukseen huonoista sivuista. (Nielsen 2011)

6.3 JavaScript ja HTML

JavaScript on web-ympäristöön tehty skriptikieli. Se mahdollistaa erilaisien toiminnallisuuksia lisäämisen sivuille, kuten tekstin suurentamisen napista. Toisin kuin Javaa, JavaScriptiä ei kuitenkaan tarvitse kääntää minkäänlaiseksi tavukoodiksi ennen ajamista, vaan ohjelma kirjoitetaan suoraan www-sivulle. Näin ei tarvitse käyttää erillistä kehitysympäristöä skriptin tekemiseen. (Särkkä 1998)

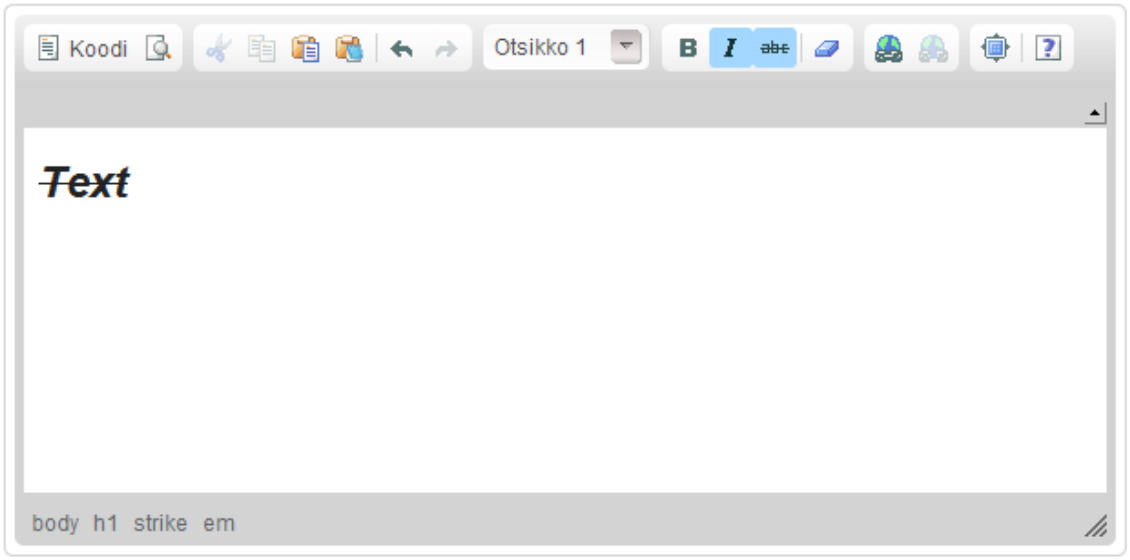
HTML:n (HyperText Markup Language) tehtävänä on kuvata dokumentin tekstin *rakenne*, ei niinkään ulkoasua. WWW-palvelimesta sivun hakenut selainohjelma päättää asetustensa perusteella, miten teksti esitetään käyttäjän silmien edessä.

6.4 JQuery ja CKEditor

JQuery on JavaScript-kirjasto, joka lisää ominaisuuksia JavaScriptiin. Ominaisuuksia on lukuisia ja suurin osa lisää helppokäyttöisyyttä web-käyttöliittymään. Kuten esim. päivän valitseminen kalenterista, jotta käyttäjän ei tarvitse kirjoittaa sitä lomakkeessa. JQueryyn on myös saatavilla erittäin paljon erilaisia liitännäisiä (plugineja), jotka lisäävät paljon niin ulkoasuun kuin toiminnallisuuteen tarvittavia ominaisuuksia.

Käytän myös ohjelmassa CKEditoria. Kyseinen editori mahdollistaa HTML–koodin käsittelyn helposti selaimella. Editorilla on tarkoitus muokata tietokantaan tallennettuja sähköpostiviestien malleja, jolloin ei tarvitse käyttää erillistä

sähköpostisovellusta viestien lähettämiseen. Kyseessä on vapaaseen lähdekoodiin perustuva editori, joten siksi valitsin sen ohjelmaani. Kuva 4 näyttää miltä CKEeditori näyttää selaimessa.



Kuva 4. CKEditori mahdollistaa tekstin monenlaisen muokkauksen

6.5 MySQL

MySQL on relaatiotietokanta. Tietokanta on varsin tehokas tapa tallentaa niin suuria kuin pieniä tietomääriä. Tiedon hakeminen tietokannasta on erittäin helppoa ja sitä voidaan hakea hyvin monimutkaisilla ehdoilla. Tietokanta koostuu tauluista, jotka sisältävät joukon tietueita, joilla on ennalta määritelty rakenne. Yhdessä taulussa voi olla vaikka varaston tavaroiden tiedot, jolloin yhtä tavaraa vastaa yksi tietue. (Ohjelmointiputka 2011)

Ajanvarauksessa voidaan tehdä hyvinkin monimutkainen tietokanta. Tietokannan kokoon vaikuttaa suurimmaksi osaksi se, kuinka monipuolinen ohjelman pitää olla, esim. pitääkö ohjelmaan kirjautua ja mitä tietoja ohjelman haltija tarvitsee varaajalta. Yleensä tarvittavia tietoja ovat nimi, osoite, puhelinnumero ja sähköpostiosoite.

Mikäli halutaan muokata jokaista aikaa erikseen, täytyy ne tallentaa tietokantaan. Tällöin tietokanta vie paljon tilaa, jolloin tietokannan optimointi voi olla hy-

vin tärkeää, koska suurista tietomääristä on hitaampaa hakea tietoa. Mikäli tietokanta on normalisoitu oikein, toimii se hyvin isoimmillakin tietomäärillä. Pitää kuitenkin muistaa, että liika normalisointi voi hidastaa tietokantaa.

Tein ajanvarauksen prototyyppiin yksinkertaisen tietokantamallin. Jokainen vapaa aika tallennetaan tietokantaan, koska piti olla myös mahdollista muokata tai poistaa aikoja. Prototyypissä suurin ongelma oli varaajien yksilöinti, koska varaajan sitominen käyttäjätunnukseen ei tule kysymykseen. Tämä siksi, että ohjelmaan ei tarvitse erikseen kirjautua. Mikäli tietokantaan tulee kaksi aivan samannimistä, ei voida tietää nimen mukaan, onko kyseessä sama vai eri ihminen. Tällöin tärkeiksi tulevat varaajan muut tiedot, kuten esim. sähköpostiosoite tai ryhmätunnus.

Ohjelmassa pitää olla myös mahdollisuus peruuttaa aika ajan varaajan toimesta. Tämän toteutin siten, että jokaiseen varaukseen generoidaan 10-numeroinen luku, joka lähetetään ajan varaajalle sähköpostilla. Ajan varaaja syöttää peruutuslomakkeeseen luvun ja varmistuksen jälkeen varaus poistetaan tietokannasta. Peruutuskoodin piti olla mahdollisimman pitkä, jotta kuka tahansa ei voi keksiä niitä ja peruuttaa toisten aikoja. Mitä suurempi luku sitä enemmän se vie tilaa tietokannassa, joten se ei saisi myöskään olla liian pitkä. Tästä syystä päädyin 10-numeroiseen lukuun.

Muut tallennettavat tiedot ovat etunimi, sukunimi, ryhmätunnus ja sähköposti. Sähköpostiosoitteeseen lähetetään varmistusviesti varauksen jälkeen. Viestissä ilmoitetaan aika ja peruutuskoodi. Sähköpostin lähetys onnistuu helposti käyttämällä PHP:n mail()-funktiota. Mail()-funktio on PHP:n omia valmiita funktioita ja se lisättiin versiossa 4.

Ajanvarauksen hallintaa varten pitää tallentaa sähköpostimallit ajan peruuttamisesta, siirtämisestä ja poistamisesta. Mikäli esim. opettaja joutuu peruuttamaan ajan, peruutuksen yhteydessä kysytään lähetetäänkö varaajalle asiasta sähköpostiviesti. Mikäli valitaan sähköpostiviestin lähetys, ilmestyy Internet-lomake, jossa on aiemmin kertomani CKEditori ja editori sisältää sähköpostiviestin mal-

lin. Mallin tarkoitus on nopeuttaa viestien lähetystä, koska tällöin ei tarvitse joka kerta kirjoittaa viestiä uudestaan, vaan se on ohjelmassa valmiina.

Tietokantaa voidaan käyttää silloin, kun halutaan sivuista monikielisiä. Yksinkertaisesti tietokantaan tallennetaan eri kielellä asiat ja haetaan tietoja sitä mukaan, kun sivuja selaillaan. Prototyypin perusteella en itse käyttänyt ko. tapaa, sillä varaajan sivut ovat hyvin pienet. Helpoiten päästään tekemällä kaksi eri sivua suomenkielisille ja englanninkielisille. Suurissa sivustoissa kannattaa käyttää tietokantoja tai muuta tapaa, sillä sivujen tekeminen erikseen jokaiselle kielelle ei ole kannattavaa, koska niiden ylläpito on työlästä.

7 OHJELMAN POHJATYÖ

Pohjatyönä aloin tutustumaan erilaisiin jo valmiina oleviin Internetissä oleviin ilmaisiin ajanvarausohjelmiin. Heti ensimmäiseksi kiinnitin huomiota miten ohjelmissa itse ajanvaraus etenee. Ajanvaraamisessa on oikeastaan yksi loogi-nen etenemisreitti, joka on seuraavanlainen:

1. Valitaan päivä. (Tässä kohtaa voidaan myös valita ensin varattava paikka/toimipiste/huone/työntekijä ja sitten vasta päivä, riippuen yrityksestä.)
2. Valitaan listasta vapaana oleva aika tai syötetään aika sille varattuun kenttään.
3. Ilmoitetaan varaajan tiedot. Voidaan kysyä mm.
    - nimi
    - osoite
    - puhelinnumero
    - sähköpostiosoite.
4. Lopuksi ohjelma ilmoittaa onnistuiko varaus. Kehittyneemmät ohjelmat voivat ilmoittaa vielä varauksen tiedot tekstiviestillä tai sähköpostilla varaajalle.

Suurimmassa osassa ohjelmista oli valmiina jonkinlainen kalenteri, missä voidaan valita helposti päivä ja tämän jälkeen tulostetaan vapaat ajat. Joissakin tapauksissa ajalle oli annettu vain tekstikenttä, johon aika kirjoitettiin muodossa pp.kk.vvvv.

Ohjelman antamia virheilmoituksia ei voinut testata kovinkaan paljoa, joten ne rajoittuivat virheellisen tiedon syöttämiseen lomakkeella. Ohjelmat antoivatkin hyvin palautetta virheellisistä syötteistä. Ohjelman ilmoittamien varoituksien ja ilmoituksien miettiminen etukäteen on hyvin tärkeää, sillä varoituksien ei pidä olla vaikeaselkoisia. Käyttäjän pitää ymmärtää virhetilanteissa ohjelman antamat virheilmoitukset ja miten virhe korjataan, esim. syötettäessä sähköpostiosoitetta osoitteesta puuttui symboli @.

Käyttäjälle pitää kertoa selkeästi ajanvarauksen onnistumisesta, sillä ilman ilmoitusta käyttäjä ei ole täysin varma onnistuiko varaus. Pahimmassa tapauksessa käyttäjä joutuu soittamaan varauspaikkaan ja kysymään onnistuiko vara-

us. Tällöin ajanvarauksen suurin tavoite, eli resurssien vapauttaminen muihin työtehtäviin ei toteudu.

7.1 Ajas Internet-ajanvaraus

Valitsin esiteltäväksi ajanvarausjärjestelmäksi Ajas Internet-ajanvarausjärjestelmän, koska se oli monipuolisin ja toimi hyvin. Kyseessä on perinteinen ajanvarausjärjestelmä, jolla voidaan varata aika itselle sopivaan aikaan ja selata vapaita aikoja. Ohjelma sisältää myös monipuolisen hallintapaneelin, jonka avulla voidaan mm. lisätä puhelimitse tehtyjä varauksia helposti järjestelmään.

Kuvasta 5 voidaan huomata, että ohjelma on ulkoasullisesti hyvin yksinkertainen ja varsin mukava käyttää. Sovelluksen hallinta jatkoi samalla pelkistetyllä linjalla. Lomakkeiden asettelu oli ohjelmassa varsin hyvä, sillä kaikki tieto on keskellä sivua, tällöin haluttu tieto löytyy yleensä nopeasti. Ohjelma myös toimi nopeasti, mikä on hyvin tärkeää. Varaamisen onnistuminen ilmoitettiin myös selvästi.

Kuva 5. Ajas Internet-ajanvarausjärjestelmä (www.ajas.fi)

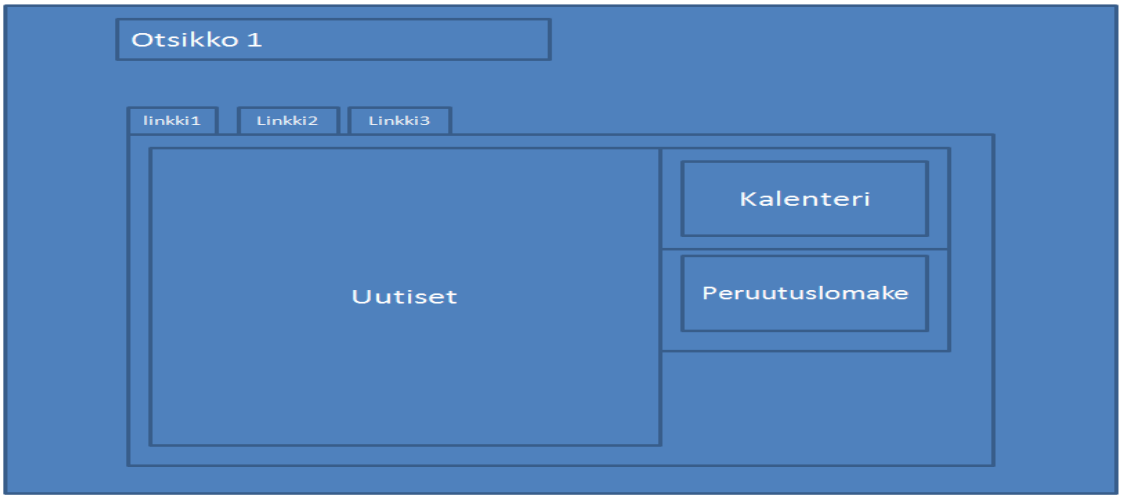
Kuvasta 5 voidaan huomata myös peruskomponentit ajanvaraukseen, kalenteri ja vapaiden aikojen tulostaminen taulukossa. Ajanvarauksessa yhdelle ajalle on varattu 15 minuuttia, mikä voi olla joissakin tapauksissa liian vähän. Tällöin va-

raajan pitää varata kaksi aikaa. Vapaat ajat ovat linkkejä, joita ohjelma etenee varaajan tietojen syöttämiseen. Tämän jälkeen varmistetaan tiedot ja tehdään ajanvaraus valittuun aikaan.

Jokaisessa ohjelmassa oli oma tapansa tulostaa tiedot, sillä Internet-tekniikat tarjoavat mahdollisuudet erillisiin ulkoasullisiin ratkaisuihin. Hyvin monessa testatussa ohjelmassa oli yksinkertainen ulkoasu. Tämä nopeuttaa itse ajanvarausprosessia, sillä yksinkertainen käyttöliittymä on yleensä helppokäyttöinen.

7.2 Ajanvarauksen ulkoasun suunnittelu

Lähdin varaussivun ulkoasun suunnittelussa siitä lähtökohdasta, että sivuja on helppo käyttää ja sivut toimivat nopeasti. Tämä tarkoittaa sitä, että sivut ovat yksinkertaiset ulkoasultaan eli sivuille ei tule mitään Flash -animaatioita tai muuta tietokoneen resursseja paljon kuluttavaa käyttöliittymää. Kuvassa 6 on yksinkertainen suunnitelma ajanvarauksen etusivusta.



Kuva 6. Yksinkertaistettu ulkoasu ajanvarauksen etusivusta

Kuvasta 6 näemme, että sivulla voidaan tehdä kaikki tarpeellinen: lukea uutisia, valita kalenterista päivä ja tehdä vapaaseen aikaan varaus, sekä tietysti peruuttaa aika. Näin sivujen käyttäjän ei tarvitse etsiä sivuilta asioita, vaan ne ovat heti etusivulla.

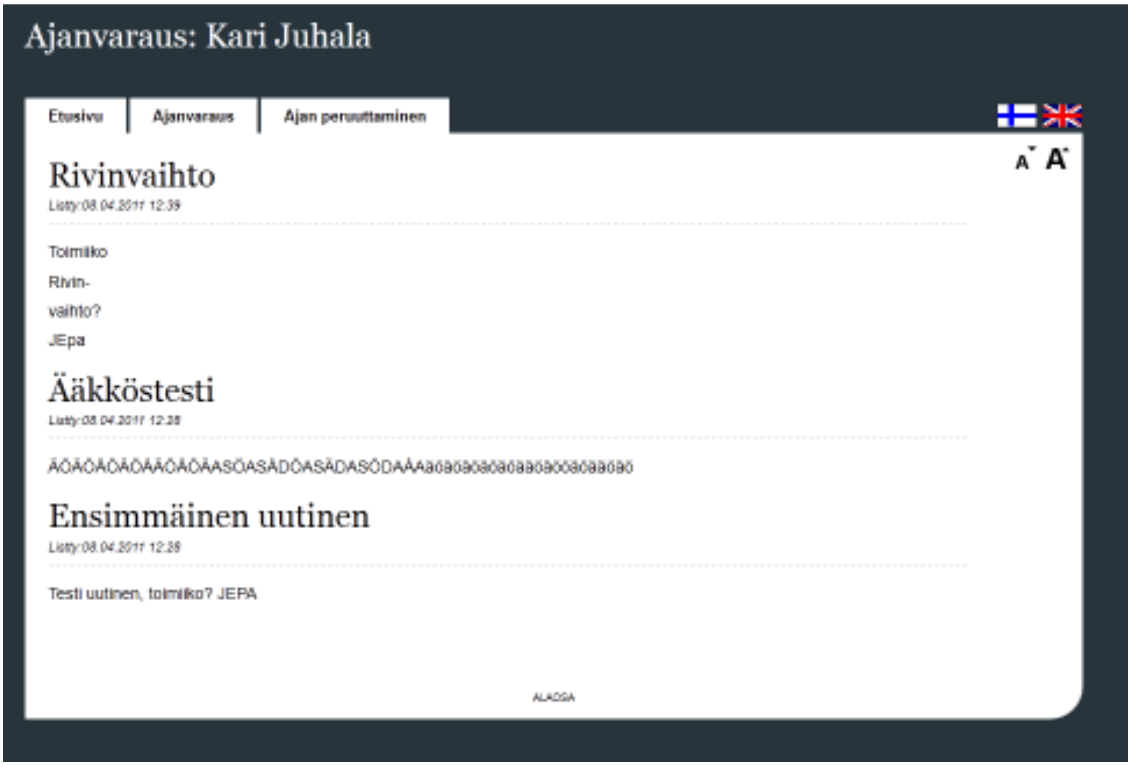
Sain tehtyä prototyyppiin kuitenkin erilaisen varattujen aikojen listauksen, jossa ei käytetä ollenkaan kuukausi kalenteria, vaan tulostetaan viikon kaikki vapaat

ajat (kuva 7). Yksinkertaisesti tein taulukon, johon listataan viikon vapaat ajat, aloitetaan maanantaista ja lopetetaan sunnuntaihin. Viikkojen välillä voidaan liikkua painamalla nuolia. Poistin kalenterin prototyypissä, sillä tämä versio tuntui nopeammalta ja helpommalta käyttää.

Ma 02.05.2011	Ti 03.05.2011	Ke 04.05.2011	To 05.05.2011	Pe 06.05.2011	La 07.05.2011	Su 08.05.2011
10:00-11:00	10:00-11:00	10:00-11:00	10:00-11:00	10:00-11:00	10:00-11:00	10:00-11:00
11:00-12:00	11:00-12:00	11:00-12:00	11:00-12:00	11:00-12:00	11:00-12:00	11:00-12:00
15:00-16:00	15:00-16:00	15:00-16:00	15:00-16:00	15:00-16:00	15:00-16:00	15:00-16:00

Kuva 7. Ajanvarauksen prototyypin aikojen selaaminen

Koska poistin prototyypistä kalenterin, täytyi myös etusivu tehdä uudestaan. Uudeksi etusivuksi suunnittelin sivun, johon listataan vain uutiset (kuva 8). Poistin sieltä myös aikojen peruutus -lomakkeen. Tämä siksi, että uutisten listaamiseen tulisi enemmän tilaa sivusuunnassa. Aikojen peruuttamiselle tein omat sivut.



Kuva 8. Ajanvarauksen prototyypin etusivu

Etusivusta voidaan myös huomata heti kielenvalinnan alapuolella (=eri maiden lippu), kaksi erikokoista A –kirjainta (kuva 8). Kyseisillä kuvalinkeillä voidaan pienentää ja suurentaa tekstiä, mikäli näin halutaan. Tekstin suurentamiseen käytetään JavaScriptiä. Lopullisesta ulkoasusta poistin kyseiset toiminnot, sillä nykyään voidaan selaimella suurentaa tekstiä helposti, joten en nähnyt ko. ominaisuutta tarpeelliseksi. Siirsin kielen vaihtamiseen tarkoitetut liput tekstin koon muokkaamisen nappien paikalle (kuva 9).

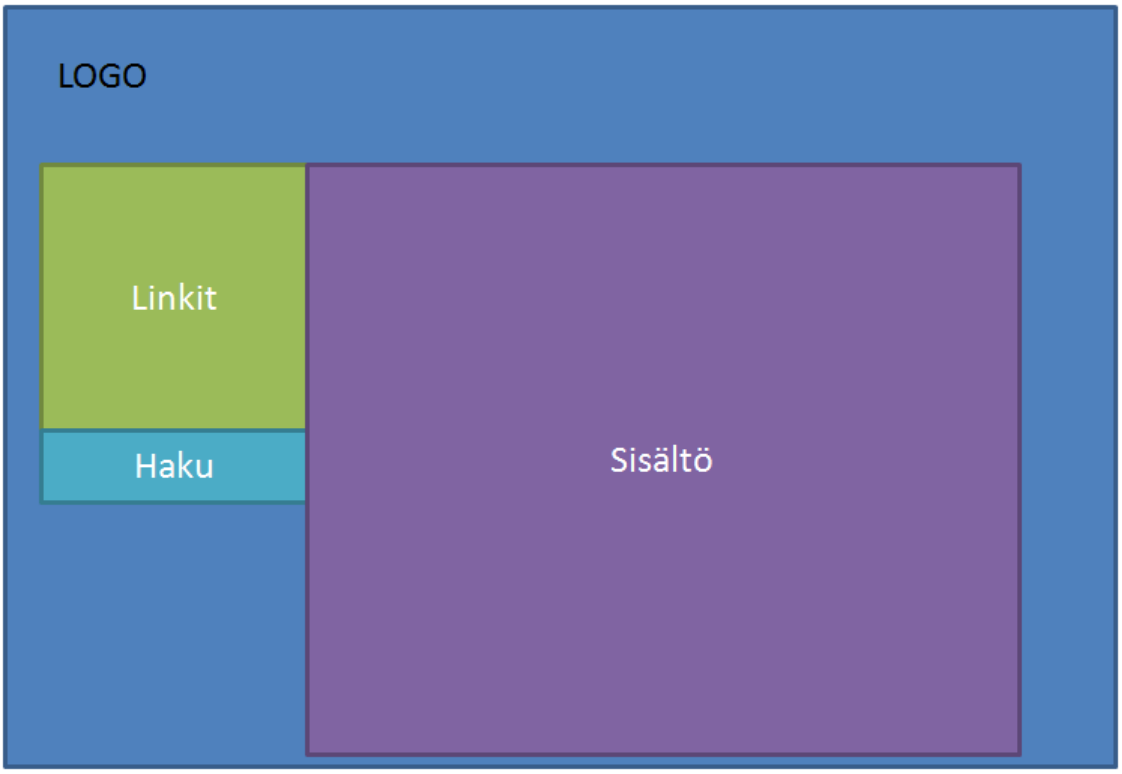
Kuva 9. Ajanvarauksen lopullinen ulkoasu

Kuvassa 9 on lopullinen versio etusivusta ja poistin siitä etusivu–linkkitekstin. Kyseisen linkkitekstin vaihdoin uutiset-linkkitekstiksi. Tämä siksi, että etusivu ei kuvannut tarpeeksi hyvin sivulla olevaa asiaa. Ulkoasussa pitää miettiä hyvin tarkkaan eri linkkien nimet, jotta ne kuvaisivat sivun sisältöä mahdollisimman hyvin. Näin vältytään siitä, että käyttäjä ei eksy sivuilla. Muuta muutosta linkkeihin ei tarvinnut tehdä lopullisessa versiossa.

Ulkoasussa voi huomata pyöristyksen oikeassa alakulmassa. Tein pyöristyksen käyttämällä CSS–komentoa, joka on käytössä vain uusimmissa selainversioissa. Mikäli selain ei tue kyseistä pyöristystä, reuna on samanlainen suorakulmainen kuin muissakin kulmissa. Pyöristys oli käytössä jo ensimmäisessä CSS–koodissani, joten jätin sen siihen lopullisessa versiossa, koska sen poistamiselle ei ollut mitään erityistä syytä. Yleensä pyöristys on tehty erikseen kuvalla.

7.3 Hallinnan ulkoasun suunnittelu

Hallinnan ulkoasun suunnitelma oli myös varsin yksinkertainen. Siinä on vain logo, linkit ja sivun sisältö. Ajatuksena oli se, että kaikki tarpeelliset linkit olisivat aina käyttäjän näkyvissä. Tällöin ohjelman käyttö olisi mahdollisimman nopeaa ja helppoa. Kuvassa 10 on yksinkertainen suunnitelma hallinnan etusivusta.



Kuva 10. Hallinnan etusivusta tehty yksinkertainen malli

Linkit ja haku ovat siis aina näkyvissä ja sisällössä näytetään halutut tiedot. Haku hakee kaikista mahdollisista tietokannan tauluista, sillä siihen ei tule erikseen valintaa mistä taulusta haetaan. Tämä yksinkertaistaa hakua, koska voidaan hakea suoraan kaikista tietokantatauluista.

Kuva 11. Hallinnan prototyypin etusivu

Hallinnan prototyyppiin ei tarvinnut tehdä juuri muutoksia, joten se on samanlainen valmiissa sovelluksessa (kuva 11). Vasemmalla ovat linkit, jotka on jaettu kategorioihin niiden sisältämien toimintojen mukaan. Kategorioiden tarkoitus on helpottaa hallinnan selaamista.

7.4 Ajanvarauksen sivurakenne

Pyrin tekemään ajanvarauksen sivurakenteesta mahdollisimman vähän eri tiedostoja sisältävän, jotta sivujen päivittäminen olisi mahdollisimman helppoa. Käyttöliittymän kieli riippuu siitä, minkä kielen käyttäjä on valinnut. Uutiset listataan valitulla kielellä, joten opettajan on kirjoitettava joko suomeksi, englanniksi tai molemmilla kielillä, mikäli haluaa että uutinen näkyy molemmilla puolilla. Ajanvarauksessa on seuraavanlainen sivurakenne:

1. Uutiset: Opettaja kirjoittaa hallinnassa uutisen joko suomeksi tai englanniksi. Uutiset listataan käyttäjän haluamalla kielellä ajanvarauksessa.
2. Varaa aika: Mennään varaamaan aika.
3. Peruuta aika: Voidaan peruuttaa aika peruutuskoodin avulla.
4. Ajanvarauksen ohje. Ajan varaaminen ja peruutus kerrottu helppolukuisesti.

Koska web-sivujen pitää olla helppokäyttöiset, eikä sivuilla saisi olla ohjeita, päätin poistaa ajanvarauksen ohjeen. Ohjeen tarkoitus olisi ollut kertoa yksinkertaisesti miten sivuja käytetään, mutta koska kyseessä on hyvin yksikertaiset sivut, eikä sivuja ei ole kovinkaan monta, päätin poistaa koko ohjeen. Hyvin harva lukee web-sivujen ohjeita, ennen kuin alkaa käyttää niitä.

7.5 Hallinnan sivurakenne

Hallinnan sivurakenne on jaettu eri kategorioihin, jotta sivujen selaaminen olisi mahdollisimman helppoa ja yksinkertaista. Hallinnan sivurakenne on seuraava:

1. Hallinta:
    - Lisää uusia aikoja: Lisätään tietokantaan uusia aikoja, joita oppilaat voivat varata.
    - Varauksen hallinta: Oppilaiden varauksien hallinta. Pystytään niin lisäämään, muokkaamaan ja poistamaan varauksia.
    - Uutisten hallinta: ajanvarauksessa näkyvien uutisten hallinta. Voidaan lisätä, muokata ja poistaa uutisia
    - Omien tietojen hallinta: opettajan tietojen muokkaaminen. Voidaan muokata niin käyttäjätunnusta ja salasanaa, joilla kirjaudutaan hallintaa, tai yhteystietoja joita käytetään varaussähköpostissa.
    - Sähköpostien hallinta: ajan varauksen, muokkauksen ja poistamisen yhteydessä lähtevien sähköpostien esimerkkiviestien muokkaaminen.
2. Selaaminen:
    - Viikkokalenteri: kalenteri, jossa listataan viikon varaukset päivän ja ajan mukaan
3. Tietokannan hallinta:
    - Varmuuskopiointi: tietokannan varmuuskopiointi. Voidaan viedä tai tuoda tietoa tietokannasta. Tieto on csv-tiedostomuodossa ja se mahdollistaa tiedon muokkaamisen taulukkolaskentaohjelmalla, kuten Excelillä.
    - Tietokannan tyhjennys: voidaan tyhjentää tietokanta, joko yksi taulu tai koko tietokanta kokonaan.

7.6 Uutisen lisääminen -käyttötapauksen tarkempi kuvaus

Uutisen lisääminen on varsin yksikertainen käyttötapaus ja se löytyy liitteestä 1 kokonaisuudessaan. Opettaja kirjoittaa hallinnassa uutisen, jonka haluaa lisätä ajanvarauksen etusivulle. Ohjelmissa voi olla hyvin paljon eri käyttäjiä, joten käyttötapauksen tekijät pitää erikseen ilmoittaa käyttötapausta suunniteltaessa. Tässä tapauksessa opettaja on ainut käyttäjä, joka pystyy lisäämään käyttäjän.

Esiehtojen tarkoituksena on kertoa mitä pitää ensin tehdä, ennen kuin voidaan tehdä kerrottava käyttötapaus. Käyttötapauksessa voi olla hyvin erilaisia esiehtoja, kuten esim. pitää ensin valita tietty asia valikosta, tai kirjoittaa lomakkeeseen tietty asia. Uutisen lisäämisessä pitää kirjautua hallintaan, ennen kuin opettaja voi tehdä sen, koska uutisia voidaan lisätä vain hallinnan kautta.

Kuvauksessa kerrotaan lyhyesti ja ytimekkäästi mitä käyttötapauksessa tapahtuu. Joissakin tapauksissa joudutaan kirjoittamaan hyvinkin monimutkaisesta käyttötapauksesta. Monimutkainen käyttötapaus pitää avata selkokielisesti, jotta tiedetään varmasti, minkälainen käyttötapaus on kyseessä.

Käyttötapauksen aikana tulee väistämättä poikkeuksia. Tässä tapauksessa poikkeuksena on, ”tietokannan avaaminen ei onnistu”. Tällä tarkoitetaan sitä, että uutisen tallentaminen tietokantaan ei onnistu, koska ei ole saatu yhteyttä tietokantaan. Yhteyden muodostaminen tietokantaan on monimutkainen tapahtuma, sillä ongelmana voi olla Internet-yhteyden katoaminen, ongelma koodissa tai palvelimen kaatuminen. Ongelman korjaaminen nopeasti on vaikeaa, joten siihen pitää varautua. Mikäli tietokantaan ei saada yhteyttä, koko ohjelma ei toimi lainkaan.

7.7 Oppilaan käyttötapaukset

Oppilaalle ei ole kovinkaan monta käyttötapausta ja kaikki käyttötapaukset oppilaasta löytyy liitteestä 1. Aikojen varaaminen ja peruuttaminen ovat suurimmat käyttötapaukset. Käyttötapauksiin ei tarvinnut myöskään tehdä kovin paljon muutoksia lopullisessa versiossa. Ehkä huomattavin oli Aikavaraus, joka on yk-

sinkertaistettu lopullisessa ohjelmassa. Valmiissa sovelluksessa ei tarvitse painaa varaa -nappia. Tämä siksi, että suunniteltua kuukausikalenteria tehty, vaan sen korvasi viikkokalenteri, jossa on viikon ajat heti näkyvillä. Suunnitellut varaustiedot olivat riittävät ja niitä ei tarvinnut muokata (etu- ja sukunimi, sähköpostiosoite, ryhmätunnus ja tapaamisen aihe). Valmis ohjelma lähettää opettajan tallentaman sähköpostipohjan mukaisen viestin varaajan sähköpostiin.

Koska ohjelmassa ei ole mahdollisuutta kirjautua, ei oppilas voi muokata varaustaan. Muokkauksen olisi tietysti voinut tehdä peruutuskoodin avulla, mutta tällöin tulee ongelmaksi mitä tietoja oppilaan annetaan muokata. Mikäli hänen annetaan muokata vain yhteystietojaan, ei niiden muokkaus ole järkevää, koska niiden muutoksella ei ole vaikutusta itse varaukseen ja tuskin kovinkaan monen tarvitsee muokata niitä. Mietin myös sitä, että käyttäjä annetaan muokata aikaa, mutta näin ominaisuuden turhaksi, sillä saman voi tehdä peruuttamalla ajan ja varaamalla uuden ajan. Siksi en antanut oppilaalle mahdollisuutta muokata varaustaan.

7.8 Opettajan käyttötapaukset

Opettajan käyttötapaukset löytyvät kokonaisuudessaan liitteesta 1. Koska opettajan pitää hallita koko sovellusta ja saada sieltä tarvitsemaansa tietoa, on hänellä paljon enemmän käyttötapauksia oppilaaseen verrattuna. Opettajan käyttötapaukset muuttuivat joissakin kohdissa varsin paljon, sillä vasta prototyyppiä tehtäessä tuli huomattua, että jotkin toiminnot kannattaa tehdä toisella tavalla tai toiminnallisuutta piti lisätä . Ehkä paras esimerkki oli uutisen lisääminen, sillä siihen tuli mahdollisuus lisätä suomen- ja englanninkielisiä uutisia. Myös ajan peruutuksessa ja siirrossa voidaan lähettää viesti varaajan sähköpostiin, jolloin opettajan ei tarvitse erikseen lähettää ajan peruutuksesta tai siirrosta sähköpostia, vaan se onnistuu ohjelman kautta nopeasti.

Sähköpostin lähetys varaajalle tuli siis niin varatun ajan siirtämiselle ja ajan poistamiselle. Tämän tarkoituksena on se, että varaaja saa varmasti tiedon peruutuksesta tai siirrosta. Ajanvarauksessa tiedon saanti, niin opettajan kuin oppilaan, on erittäin tärkeää. Mikäli tieto ei kulje, voi opettaja odottaa turhaan oppi-

lasta tulevaksi paikalle, mutta toimii myös toisinkin päin. Tästä syystä ohjelmaan piti lisätä sähköpostinlähetys.

Tietokannan hakeminen xls–tiedostoksi mahdollistaa tietokannan tutkimisen Excel -taulukkolaskennassa. Ominaisuuden tarkoituksena on mahdollistaa se, että mikäli ei ole Internet-yhteyttä voidaan silti katsoa päivän varaukset. Tietysti varauksien tiedot ovat vanhoja ja mikäli joku on perunut ajan tiedoston ottamisen jälkeen, ei sitä näy. Joissakin tapauksissa kyseessä on varsin käytännöllinen ominaisuus. Tietokantaan voidaan myös tallentaa uusia aikoja kirjoittamalla ne Excel–tiedostoon. Tällöin aikojen lisäämiseen ei tarvitse käyttää sille tarkoitettua sovellusta hallinnassa.

Jotta tietokanta pysyisi mahdollisimman pienenä, pitää siinä olla vanhojen aikojen ja varauksien poistaminen. Tällöin tarkistetaan kaikki ajat, ja mikäli aika on vanhempi kuin meneillään oleva, se poistetaan. Ajalle tehty varaus poistetaan myös ajan poiston yhteydessä. Tietokannan puhdistaminen vanhoista ajoista ja varauksista tekee myös tietokannasta hakemisen nopeammaksi, koska tällöin on vähemmän selvitettävää.

8 TIETOKANTALUOKAN ESITTELY

Tietokantaluokka löytyy kokonaisuudessaan liitteestä 2. Jotta tietokannasta hakeminen ja sinne tallentaminen olisi mahdollisimman helppoa, tein siihen erillisen tietokantaluokan. Kun tietokantaluokan avulla haetaan tietoja, se antaa kaksiulotteisen taulukon, jossa on kyselyn tulokset. PHP:ssä kaksiulotteinen taulukko on tyyliltään $taulukko[][]. Tietokantaluokassa ensimmäinen hakasulje kertoo monesko rivi on kyseessä ja toinen hakasulje sisältää tarvittavan tiedon, mitä ollaan hakemassa. Esimerkiksi, jos haetaan SQL-lauseella varausta jonka varausID on 42, saadaan ensimmäinen etunimi tulostamalla $taulukko[0][”enimi”]. Koska tuloksia on vain yksi, on ensimmäisessä hakasulkeessa 0 (taulukko alkaa 0:sta). Mikäli SQL–lauseella tulee paljon tuloksia, mennään taulukko läpi perinteisellä for–lauseella.

Tietokantaluokalla otetaan seuraavalla tavalla yhteys:

1. Luodaan olio, esim. $db = new tietokanta();
2. Asetetaan yhteystiedot oliolle, $db->aseta_yhteystiedot("palvelin","kayttajatunnus","salasana","tietokannan nimi");
3. Yhdistetään annetuilla tiedoilla tietokantapalvelimeen, $db->yhdista();
4. Tehdään kysely, $db->kysely("Select * from Kieli WHERE Kieli='Suomi'");
5. Tarkistetaan tuliko haulla tuloksia, if($db->kyselyn_rivimaara()>0)
6. Mikäli kyselyn_rivimaara antaa tulokseksi enemmän kuin 0, laitetaan tulokset kaksiulotteiseen taulukkoon, $kieli_id=$db->kyselyn_rivit();
7. Mikäli halutaan tulostaa kaikki tulokset käytetään for –lausetta.

8.1 Tietokantaluokan esittely

Käyn seuraavaksi läpi tietokantaluokan lähdekoodin. Koko tietokantaluokan lähdekoodi on luettavana opinnäytteen liitteessä 2.

```
class tietokanta
{
		var $palvelin;
		var $kayttaja;
		var $salasana;
		var $tietokanta;

		var $rivit;
		var $tulokset;
		var $yhteys;
		var $valmius;
```

Aluksi luodaan tietokantaluokka ja esitellään siinä käytettävät muuttujat. Tässä tapauksessa tarvitaan:

1. Palvelin: palvelimen osoite
2. Käyttäjä: käyttäjätunnus
3. Salasana: salasana
4. Tietokanta: yhdistettävän tietokannan nimi
5. Rivit: kyselyn tiedot, mikäli kysely antaa tulokset.
6. Tulokset: kyselyn tulokset, tarkistetaan onko kyselyllä yhtään tulosta.
7. Yhteys: yhdistetään tietokantaan
8. Valmius: voidaan tarkistaa onko yhteys muodostettu tietokantaan

```
public function __construct()
{
		$this->tulokset = null;
		$this->valmius = false;

}
```

Kun luokka luodaan, voidaan konstruktorilla antaa luokan tietojäsenille järkevät alkuarvot.

```
public function __destruct(){ @mysql_close($this->yhteys); }
```

Oliota tuhottaessa ajetaan destructori, jolloin varmistutaan, että tietokantayhteys katkeaa, eikä jää taustalle viemään palvelimen resursseja.

```
public function setPalvelin($palvelin){ $this->palvelin = $palvelin; }
public function setKayttaja($kayttaja){ $this->kayttaja = $kayttaja; }
public function setSalasana($salasana){ $this->salasana = $salasana; }
public function setTietokanta($tietokanta){ $this->tietokanta = $tietokanta; }
```

Setterien avulla annetaan metodeille tarvittavat tiedot, jotta ohjelma toimisi oikein. Tässä tapauksessa annetaan tarvittavat tiedot, joiden avulla ohjelma yhdistyy tietokantaan.

```
public function aseta_yhteystiedot ($palvelin=null, $kayttaja=null, $salasana=null, $tietokanta=null)
{
        if(!isset($palvelin,$kayttaja,$salasana,$tietokanta))
                die("Tarkista yhteyden tiedot");
        $this->setPalvelin($palvelin);
        $this->setKayttaja($kayttaja);
        $this->setSalasana($salasana);
        $this->setTietokanta($tietokanta);
        $this->isReady = true;
}
```

Aseta_yhteystiedot -funktio asetta tarvittavat tiedot settereiden avulla, mikäli funktio ei saa tarpeeksi tietoja on tarvittavien muuttujien oletusarvoksi annettu NULL, eli tyhjä. Myös muuttuja isReady -arvo muutetaan todeksi, tämän avulla voidaan tarkistaa onko tietokantaan saatu yhteys, kun kutsutaan yhdistä -funktiota.

```
public function yhdista()
{
        if(!$this->isReady) die("not ready to connect ".mysql_error($this->yhteys));
        $this->yhteys = mysql_connect($this->palvelin,$this->kayttaja,$this->salasana) or die("Ei voitu yhdistää tietokantaan ".mysql_error($this->yhteys));
        $this->kanta($this->tietokanta);

}
public function kanta($tietokanta)
{
        $this->setTietokanta($tietokanta);
        mysql_select_db($this->tietokanta,$this->yhteys) or  die ("Tietokantaan ei löydy.".mysql_error($this->yhteys));
}
```

Kun kaikki tarvittavat tiedot on annettu edellisessä funktiossa, kutsutaan funktiota yhdistä. Tässä tarkistetaan onko isReady true, mikäli tämä toteutuu, yhdistetään annettuun palvelimeen. Virhetilanteissa annetaan virheilmoitus. Samalla kutsutaan kanta–funktiota, joka ottaa yhteyden haluttuun tietokantaan.

```
public function kysely($sql)
{
		$this->tulokset = mysql_query($sql,$this->yhteys) or die
("Virheellinen kysely.".mysql_error($this->yhteys));
}
```

Kun tietokannasta haetaan tietoa, tehdään se kutsumalla kysely –funktiota. Kysely tehdään seuraavalla tavalla: $db->kysely("Select * from Kieli WHERE Kieli='Suomi'"). Tulokset tallennetaan tulokset–muuttujaan, mutta tietoja ei voida vielä tulostaa näytölle. Mikäli halutaan tallentaa tietoa tietokantaan, tarvitsee vain kutsutaan tätä funktiota.

```
public function kyselyn_rivimaara($selectMode = true)
{
		if($selectMode)
		{
			return mysql_num_rows($this->tulokset);
		}
		else
		{
			return mysql_affected_rows($this->yhteys);
		}
}
```

Kyselyn_rivimaara -funktio tarkistaa onko kysely–funktio saanut yhtään riviä, kun haetaan tietokannasta. Tämän avulla voidaan tehdä helppo if-lause, jonka avulla tarkistetaan tuliko tietoja ja mikäli ei tullut ilmoitetaan siitä käyttäjälle, muuten tulostetaan tiedot.

```
public function kyselyn_rivit()
{
		if(!$this->tulokset) die("Nothing found!".mysql_error($this->yhteys));
		$this->rivit = array();
		while($r = mysql_fetch_array($this->tulokset,MYSQL_ASSOC))
		{
			$this->rivit[] = $r;
		}
		mysql_free_result($this->tulokset);
		return $this->rivit;
}
```

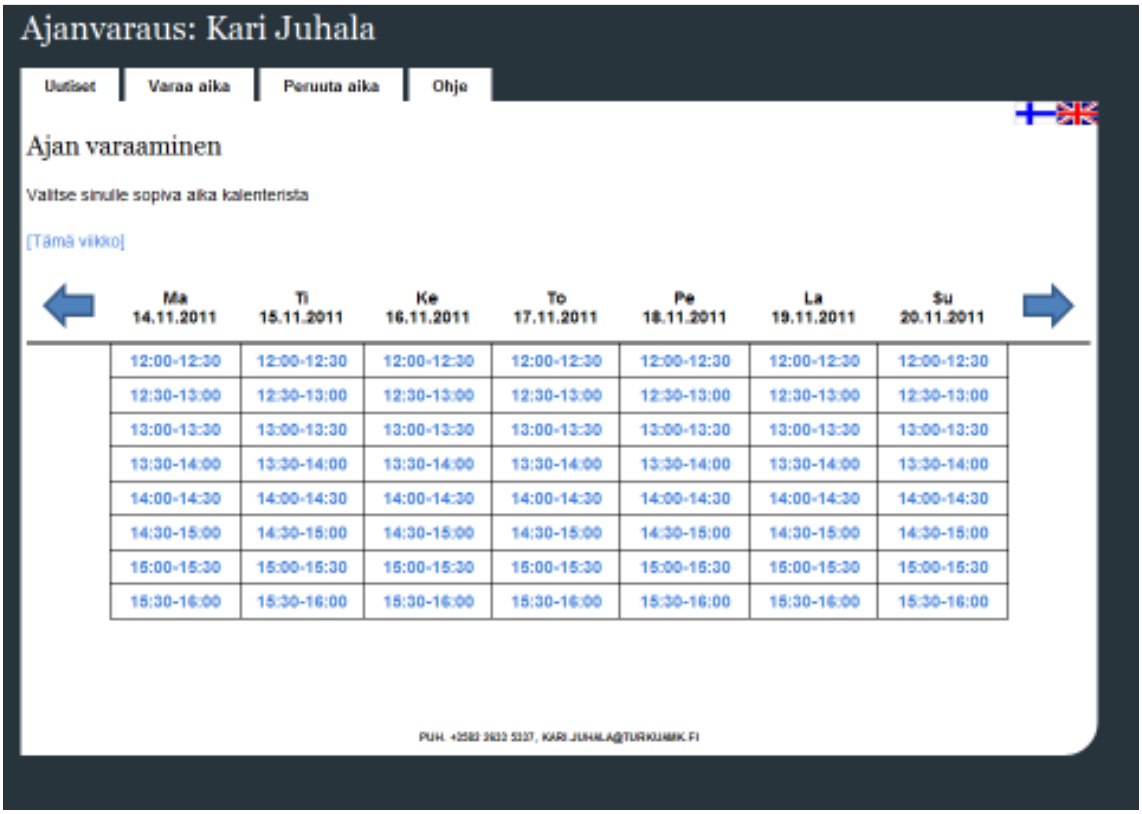
Kun on varmistuttu, että SQL–lause saa tuloksia, mennään kaikki SQL–lauseen saamat tulokset läpi ja tallennetaan kaksiulotteiseen taulukkoon.

```
public function yhteyden_tarkistus()
{
		if($this->yhteys)
		{
			return "Yhdistetty";
		}
		else
		{
			return "Ei yhdistetty";
		}
}
public function sulje()
{
		mysql_close($this->yhteys);
		$this->yhteys = null;
}
```

Yhteyden_tarkistus –funktiolla voidaan tarkistaa onko saatu yhteys tietokantaa. Sulje–funktiolla suljetaan yhteys tietokantaan, jolloin se ei jää viemään resurs-seja palvelimelta. Tämä on viimeinen funktio tietokantaluokassa.

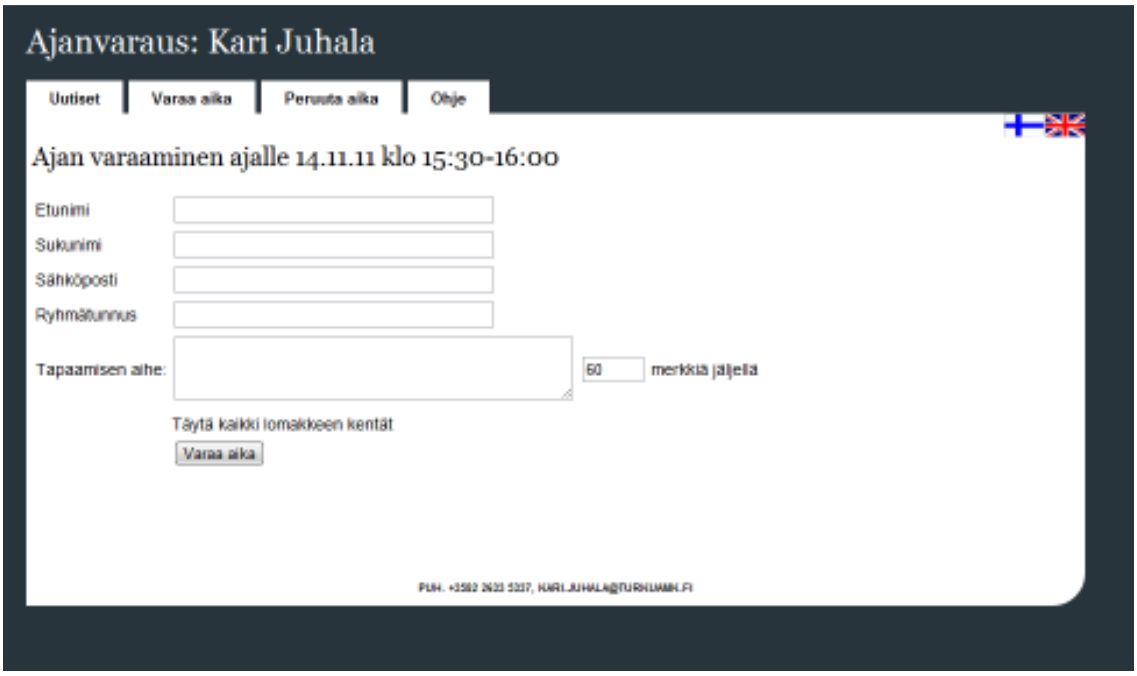
9 AJAN VARAAMINEN VALMIISSA OHJELMASSA

Ajan varaaminen alkaa, kun varaaja tulee varaussivulle. Varaussivun etusivulle tulostetaan uutiset ja siellä voidaan vaihtaa ohjelman kieli suomeksi tai englanniksi. Kun etusivulta valitaan ”Varaa aika”, tulostettaan viikon vapaat ajat kuvan 12 mukaisesti.



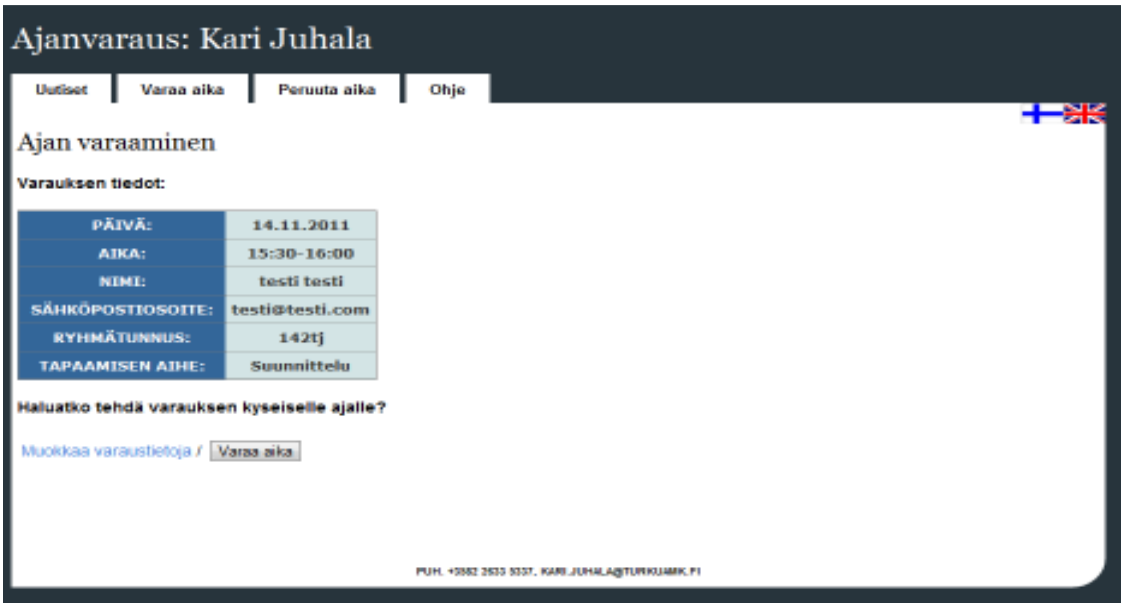
Kuva 12. Vapaiden aikojen listaaminen

Käyttäjä voi vaihtaa viikkoa nuolien avulla. Ajan tausta muuttaa väriä, kun kursori on sen päällä. Tällöin käyttäjä tietää varmasti valitsevansa oikean ajan.



Kuva 13. Varauksen lomake

Kun käyttäjä on valinnut haluamansa ajan, tulostetaan kuvan 13 mukainen varauslomake. Lomakkeessa kysytään etunimi, sukunimi, sähköpostiosoite, ryhmätunnus ja tapaamisen aihe. Kaikki lomakkeen kentät täytyy täyttää ja niiden oikeellisuus tarkistetaan. Mikäli tieto on väärin kirjoitettu, esim. sähköpostista puuttuu @-merkki tai kenttä on tyhjä, ilmoitetaan ongelma ja varaajan on korjattava tilanne, jotta varausprosessi etenee.



Kuva 14. Varaustietojen tulostaminen

Lomakkeen täytön jälkeen tulostetaan syötetyt tiedot (kuva 14). Tällöin varmistetaan, että kaikki tiedot eivät ole virheellisiä. Mikäli jokin tieto on virheellinen, tarvitsee painaa muokkaa varaustietoja -linkkiä, jolloin palataan varauslomakkeeseen ja tällöin voidaan korjata virheellinen tieto.

Kuva 15. Varaamisen loppuilmoitus

Kun kaikki tiedot on varmistettu, tulee seuraavaksi kuvassa 15 oleva ilmoitus varauksen onnistumisesta. Varaus epäonnistuu, jos joku muu käyttäjä on varannut ajan nopeammin. Tämän tarkoituksena on estää useamman varauksen tekemisen yhdelle ajalle.

10 PROJEKTIN SUURIMMAT ONGELMAT

Jokaisessa projektissa on omat ongelmansa, niin myös tässäkin projektissa. Ehkä suurimpana ongelmana oli ajan hallinta. Ajanhallinta on erittäin hankalaa, kun ei tiedä tarkalleen miten ohjelma kannattaa tehdä ja mitä tarvitsee ottaa huomioon. Tällöin aikaa kuluu paljon turhien asioiden miettimiseen ja kokeilemiseen. Myös valitsemani protoilumalli vei paljon aikaa, koska oli paljon asioita joita piti ensin kokeilla ja sitten vasta miettiä, kannattaako kokeilu lisätä lopulliseen ohjelmaan vai muokataanko sitä vielä. Kun projektia tekee yksin, ei ole kukaan sanomassa, että jokin asia kannattaa tehdä toisella tavalla.

Koska ohjelman vaatimukset eivät olleet kovinkaan tarkkoja, piti vielä miettiä kaikkia mahdollisia ominaisuuksia pelkän ajanvarauksen lisäksi. Tällöin lisäsin vaatimuksiin mm. uutisten kirjoittamisen kahdelle kielelle ja tietokannan varmuuskopioinnin, sillä en halunnut lisätä tietokannanhoitajien työmäärää. Uutisten lisääminen englanniksi ja suomeksi lisäsin, koska aikaa voidaan tilata kummallakin kielellä, joten tällöin uutistenkin pitäisi olla samalla kielellä. Tällöin varmistutaan siitä, että kaikki ymmärtävät uutiset varmasti.

Olin laskenut, että voisin käyttää protoilumallissa tekemääni ohjelmointikoodia, mutta koodi osoittautui erittäin huonoksi ja hitaaksi. Tästä syystä jouduin tekemään paljon uutta koodia ja se osoittautui paljon nopeammaksi, sillä se käytti paljon enemmän PHP:n omia funktioita ja oli paremmin luettavaa. Onneksi ohjelmassa ei ole kovinkaan monimutkaisia toimintoja, joten uuden koodin tekeminen oli helppoa.

Myös tämän opinnäytteen kirjoittaminen samaan aikaan itse ohjelman tekemisen kanssa oli vaikeaa, sillä pidin itse ohjelman tekemistä tärkeämpänä. Tietysti tämä opinnäytetyödokumentti on tärkeä, ja arvioinnissa ei oteta huomioon valmista ohjelmaa, mutta tämä opinnäyte jää kirjahyllyyn pölyttymään ja itse ohjelma toivottavasti otetaan käyttöön. Pidinkin tärkeimpänä saada valmiiksi ohjelma, joka toimii tilaajan haluamalla tavalla ja helpottaisi ajanvarausta.

11 LOPPUTULOS JA YHTEENVETO

Kuten aikaisemmin kerroin ohjelma ei valmistunut tämän opinnäytteen yhteydessä. Ohjelma ei valmistunut testauksen pidentyessä, sillä halusin ohjelmaan mahdollisimman vähän ongelmia ja ohjelmointivirheitä. Itse ohjelmasta tuli omasta mielestäni varsin onnistunut, mutta loppukäyttäjät lopulta arvioivat ohjelman onnistumisen.

Ohjelman tekemisessä tarvittiin valtavasti erilaisia tekniikoita ja osaamista, opin projektin aikana valtavasti itse ohjelmoinnista. Opin näkemään mitkä asiat ovat www-sovelluksissa tärkeitä ja mikä on eri selainten vaikutus ulkoasua suunniteltaessa. Lisäksi opin ohjelmien suunnittelua. Muutenkin kokemusta tuli vähän joka vaiheessa, sillä ensin piti ottaa asioista selvää, ennen kuin tein valintoja ohjelman tekemisessä.

Kun ohjelmointiprojektia joutuu tekemään yksin, on vaarana, kuten itselläni kävi, tarvittavan ajan laskeminen projektille. Vaarana on, että kiireessä tekee huonoja valintoja, jotka näkyvät valmiissa ohjelmassa. Ohjelmointiprojektit, koosta riippumatta, vaativat tietysti myös paljon kokemusta, sillä silloin tekijä tietää suurin piirtein miten projekti kannattaa tehdä ja millä aikatauluilla. Vielä vaikeammaksi tulevat suuret ohjelmointiprojektit, jotka vaativat valtavan määrän ymmärrystä. Vaarana on väärien valintojen lisäksi raha, sillä suuret ohjelmistot maksavat myös erittäin paljon.

Yhteenvetona voisi sanoa, että projekti oli aika raskas, koska suurin osa asioista piti tehdä prototyyppeinä, jolloin tuli tehtyä paljon turhaa. Myös tuli tehtyä aluksi paljon virheitä, jotka kostautuivat myöhemmässä vaiheessa projektia. Mikäli olisi heti tiennyt miten asioita kannattaa tehdä, olisi aikaa ohjelman tekemiseen mennyt ainakin puolet vähemmän. Kuten aikaisemmin kerroin kokemus on erittäin tärkeää ohjelmointiprojekteissa.

LÄHTEET

Agilemanifesto 2001. Ketterän ohjelmistokehityksen julistus. Viitattu 21.02.2012 http://agilemanifesto.org/iso/fi/

Ahonen, M. 2010. Tapaustutkimus: Soveltuuko Scrum vesiputousmallin korvaajaksi yrityksen sovelluskehitysprojekteihin?. Diplomityö. Elektroniikan, tietoliikenteen ja automaation tiedekunta. Espoo: Aalto-yliopisto

Barnett, R. 2008. Practical Playscript: Writing Procedure Manuals that People Can Use. 3., painos. Ngunnawal: Electronic & Database Publishing

BBC 2012, Necker Cube Experiment. Viitattu 7.3.2012 http://www.bbc.co.uk/apps/ifl/science/humanbody/mind/surveys/neckercube/decision?_next=index_1.tmpl

Boronczyk, T.; Naramore, E. & Gerner, J.; Scouarnec, Y. & Stolz, J. & Glass, M. 2009. Beginning PHP6, Apache, MySQL Web Development. Indianapolis: Wiley Publishing

Gilmore, J. 2008. Beginning PHP and MySQL From Novice to Professional. New York: Apress

Gutmans, A.; Bakken, S. & Rethans, D. 2004. PHP 5 Power Programming. Indianapolis: Prentice Hall.

Haikala, I. 2004. Ohjelmistotuotanto. Helsinki: Talentum

Haikala, I. 2011. Ohjelmistotuotannon käytännöt. Helsinki: Talentum

Kosko, B. 1993. Sumea logiikka. 3., painos. Helsinki: Art House Oy

Lehto, T. 2005. Eroon softabugeista. Viitattu 18.11.2011. http://www.tietokone.fi/lehti/tietokone_3_2005/ohjelmistojen_testaus_2534

Myers, G.; Badgett, T. & Sandler, C. Thomas, T. 2004. The art of software testing. Hoboken: John Wiley & Sons

Mäki, S. 2008. Web 2.0–verkkosovellusten piirteet ja Ajax niiden toteutuksessa. Pro Gradu. Tietojenkäsittelytieteiden laitos. Tampere: Tampereen yliopisto. Viitattu 18.11.2011. http://tutkielmat.uta.fi/tutkielma.php?id=18871

Nielsen, J. 2000. WWW-suunnittelu. Suom. Timo Haanpää. Jyväskylä: IT Press

Nielsen, J. 2011. How Long Do Users Stay on Web Pages?. Viitattu 30.1.2012 http://www.useit.com/alertbox/page-abandonment-time.html

Nixon, R. 2009. PHP, MySql & Javascript. Sebastopol:O'Reilly Media.

Ohjelmointiputka 2011. PHP-ohjelmointi: Osa 1 – Johdanto. Viitattu 18.11.2011. http://www.ohjelmointiputka.net/opas.php?tunnus=phpj

Pilone. D. & Miles, R. 2008. Head first software development. Sebastopol: O'Reilly Media.

Shore, J. & Warden, S, 2007. The Art of Agile Development. Sebastopol: O'Reilly Media.

Sommerville, I. 2011 Software Engineering - 9th ed. 9., uudistettu painos. Boston: Addison-Wesley.

Särkkä, S. 1998. JavaScript-opas. Viitattu 18.11.2011. http://users.tkk.fi/ssarkka/javascript/johdanto.html#johdanto

Wars. S. 2009. 10 Performance Tips to Speed Up PHP. Viitattu 18.11.2011. http://www.wardontheweb.com/10-performance-tips-to-speed-up-php/

LIITE 1. Vaatimusmäärittely

Dokumentin tarkoitus

Tämä dokumentti on vaatimusmäärittely ajanvarausohjelmalle, jossa on esitetty asiakkaan esittämät vaatimukset ja suppeat käyttötapaukset. Yksi dokumentin tehtävistä on toimia tarkempana kuvauksena järjestelmästä.

Tavoitteet

Tavoitteena on suunnitella ja toteuttaa WWW-pohjainen ajanvarausjärjestelmä. Järjestelmä koostuu kahdesta osasta: tietokannasta, missä varaukset sijaitsevat ja WWW-käyttöliittymästä.

Käyttöliittymän näkymät on jaettu kahteen kategoriaan, oppilas ja opettaja. Oppilas pystyy tekemään tai peruuttamaan ajanvarauksen ja selaamaan eri päivien varattavia aikoja. Opettajalla on samat näkymät kuin oppilaalla ja tämän lisäksi pystytään lisäämään, poistamaan ja muokkaamaan päivän aikoja ja tekemään erilaisia listauksia tietokannasta. Myös xml -tiedoston tekeminen tietokannasta on mahdollista sekä tietokannan varmuuskopiointi. Myös tietokannan tyhjentämisen pitää olla mahdollista. Opettajalla on myös mahdollista kirjoittaa pieniä uutisia, jotka tulostetaan ajanvarauksen etusivulla.

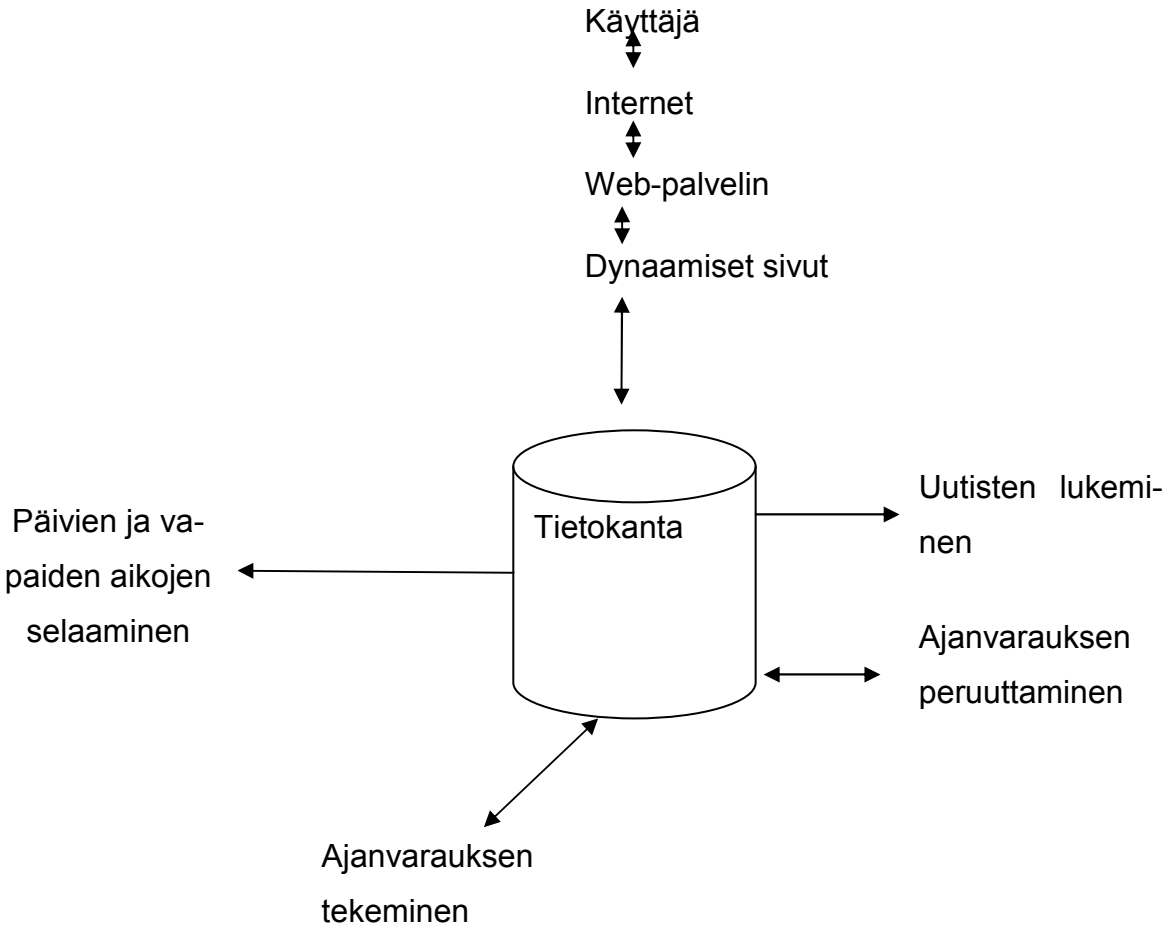
Käsitteet

Dokumentissa käytetyt termit:

Vaatimus: Toteutettavan järjestelmän toiminnallinen/ei-toiminnallinen ominaisuus tai järjestelmän rajoite.

Ei-toiminnallinen: FURPS+ periaatteen mukaan ryhmään kuuluu käytettävyys, luotettavuus, suorituskyky, tuettavuuteen, sekä ei-toiminnalliset ominaisuudet.

Järjestelmän yleiskuvaus, ajanvaraus

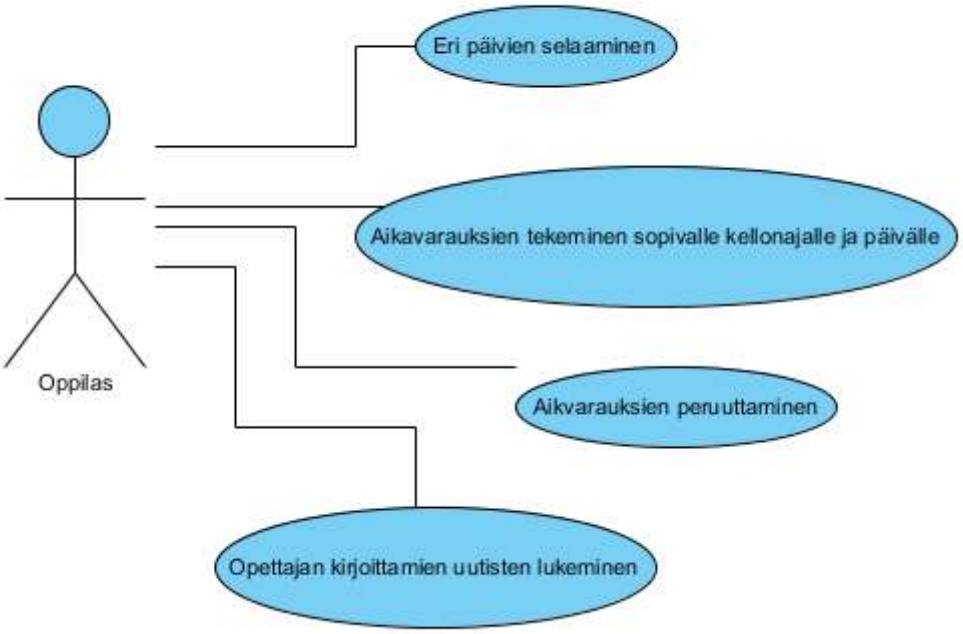


Käyttäjäryhmät

Ohjelmassa on kaksi käyttäjäryhmää: oppilas ja opettaja

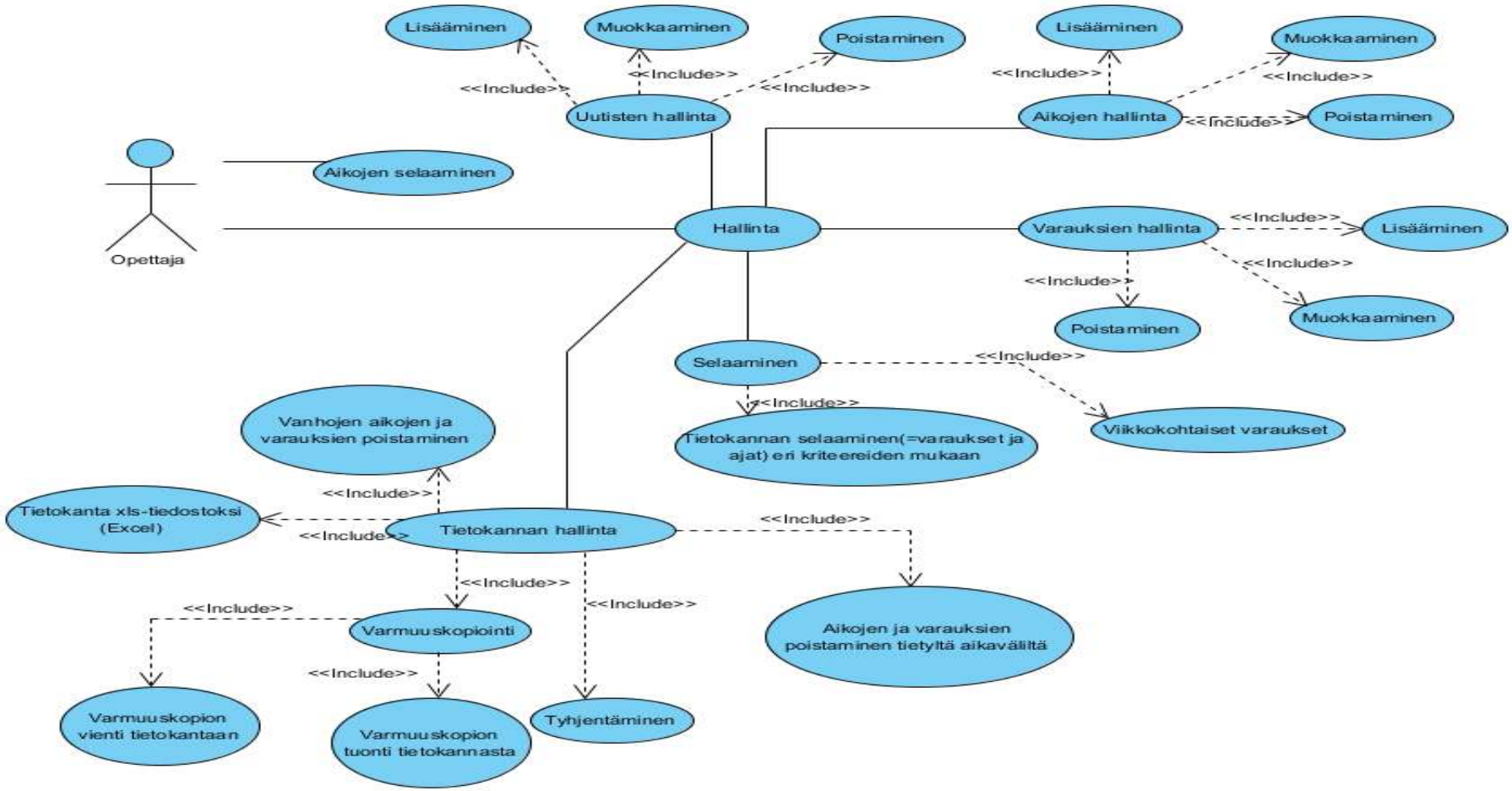
Oppilas	Oppilas pystyy selaamaan aikoja, tekemään aikavarauksen ja perumaan aikavarauksen peruutuskoodin avulla. Opettajan kirjoittamat uutiset tulostetaan ajanvarauksen etusivulla.
Opettaja	Opettaja hallitsee koko sovellusta. Eli mahdollista peruuttaa/poistaa/lisätä päivien aikoja ja varauksia sekä tekemään mm. varatuista ajoista erilaisia listauksia. Opettaja voi myös tehdä varmuuskopioita tietokannasta ja kirjoittaa uutisia.

Oppilaan käyttötapaukset



Toiminto	**Kuvaus**
Yleiset toiminnot	
Eri aikojen selaaminen	Selataan kalenterin avulla eri päiviä. Opettaja on laittanut valmiiksi päiville ajat jotka voidaan varata.
Aikavarauksen tekeminen sopivalle kellonajalle ja päivälle	Valitaan oikea päivä ja aika, painetaan varaa – linkkiä. Syötetään henkilötiedot (etunimi, sukunimi, sähköposti, luokka, tapaamisen aihe). Lopuksi oppilaalle lähetetään varauksesta ilmoitus hänen ilmoittamaansa sähköpostiosoitteeseen.
Aikavarauksen peruuttaminen	Perutaan aika syöttämällä peruutuskoodi (uniikki 10-numeroinen merkkijono)
Opettajan kirjoittamien uutisten lukeminen	Ajanvarauksen etusivulle tulostetaan opettajan kirjoittamat uutiset. Kielinä suomi ja englanti

Opettajan käyttötapaukset



Käyttötapauksien yhteenveto

Käyttötapaus: Uutisen lisääminen

Yhteenveto:	Lisätään tietokantaan uutinen
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Lomakkeen avulla lisätään uutinen tietokantaan, josta se tulostetaan kotisivun etusivulla.
Poikkeukset:	Tietokannan avaaminen ei onnistu Tallentaminen epäonnistuu
Jälkiehdot:	Uutinen tallentuu tietokantaan

Käyttötapaus: uutisen muokkaaminen

Yhteenveto:	Muokataan tallennettua uutista
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Valitaan muokattava uutinen Tulostetaan lomake, jossa on valittu uutinen
Poikkeukset:	Tietokannan avaaminen ei onnistu Tietokannan muokkaaminen epäonnistuu
Jälkiehdot:	Muutokset tallentuvat tietokantaan

Käyttötapaus: uutisen poistaminen

Yhteenveto:	Poistetaan uutinen XML- tiedostosta
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Valitaan haluttu uutinen Valitaan poista Varmistetaan poistaminen
Poikkeukset:	Tietokannan avaaminen ei onnistu Tietokannan muokkaaminen epäonnistuu
Jälkiehdot:	Uutinen poistuu tietokannasta

Käyttötapaus: Varattavan ajan lisääminen

Yhteenveto:	Lisätään aika tietokantaan, jossa se on oppilaiden varattavana
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Voidaan lisätä joko automaatin avulla tai sitten käsin jokainen aika erikseen. **Automaatti:** Kirjoitetaan aloitus-, lopetusaika ja yhden ajan pituus. Tämän jälkeen annetaan aloituspäivä ja lopetuspäivä. Viimeiseksi valitaan mille viikonpäiville aikavälissä ajat tulevat. Ohjelma tekee varattavat ajat automaattisesti aikavälille ja tallentaa ne tietokantaan **Käsin:** Jokainen varattava aika syötetään käsin (aloitu- ja lopetusaika), sekä annetaan aikaväli ja viikonpäivä mille ajat tulevat.
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe Tietokantaan tallentamisessa tapahtuu virhe Syötteet ovat virheelliset
Jälkiehdot:	Tallennetaan annetut ajat tietokantaan

Käyttötapaus: Ajan muokkaaminen

Yhteenveto:	Tallennetun ajan muokkaaminen
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Valitaan listasta haluttu aika Tulostetaan lomake jossa voidaan muokata ajan tietoja (päivä, aloitusaika, lopetusaika ja tulostetaan varaajan nimi linkkinä, mikäli ajan on varannut oppilas) Tehdään tarpeelliset muutokset ja painetaan tallenna Mikäli aika on varattu, kysytään lähetetäänkö asiasta sähköposti oppilaalle. Jos lähetetään sähköposti, tulostetaan lomake, johon voidaan kirjoittaa sähköpostin otsikko ja itse viesti.
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe Tietokantaan tallentamisessa tapahtuu virhe Syötteet ovat virheelliset Uusi aika (Aika ja kellonaika) on jo tallennettu tietokantaan. Tietokantaa ei voida tallentaa päällekkäisiä aikoja.
Jälkiehdot:	Tallennetaan muutokset tietokantaan

Käyttötapaus: Ajan poistaminen

Yhteenveto:	Tallennetun ajan poistaminen tietokannasta
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Valitaan listasta oikea päivä ja aika Varmistetaan, että ajelle ei ole tehty varausta. Mikäli ajalle on tehty varaus, tiedustellaan halutaanko se poistaa, vai vaihtaa toiselle päivälle. Mikäli ajalle ei ole tehty varausta, poistetaan se normaalisti tietokannasta.
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe Tietokantaan tallentamisessa tapahtuu virhe Syötteet ovat virheelliset Poistettavalle ajalle on tehty aikavaraus, aikavaraus joko poistetaan tai siirretään toiseen aikaan.
Jälkiehdot:	Aika poistuu tietokannasta

Käyttötapaus: Varauksen tekeminen

Yhteenveto:	Varauksen tekeminen ajalle
Toimijat:	Oppilas/opettaja
Esiehdot:	-
Kuvaus:	Listataan valitun päivän mahdolliset ajat Valitaan aika ja painetaan varaa aika Oppilaalta kysytään: Etunimi Sukunimi Sähköposti Luokka Tapaamisen aihe Tarkistetaan syötteet ja jos ne ovat kunnolliset, varaus on onnistunut
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe Tietokantaan tallentamisessa tapahtuu virhe Syötteet ovat virheelliset Poistettavalle ajalle on tehty aikavaraus, aikavaraus joko poistetaan tai siirretään toiseen aikaan.
Jälkiehdot:	Varaus tallentuu tietokantaa ja varmistus sähköposti lähetetään varaajan ilmoittamaan sähköpostiosoitteeseen

Käyttötapaus: Varauksen muokkaaminen

Yhteenveto:	Varauksen tietojen muokkaaminen
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Listataan varaukset ja valitaan oikea varaus Tulostetaan lomake, jossa on varauksen tiedot ja jossa voidaan muokata sitä Muokattavat tiedot: Etunimi Sukunimi Sähköposti Luokka Peruutuskoodi Tapaamisen aihe Tehdään tarvittavat muutokset lomakkeelle ja tallennetaan
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe Tietokantaan tallentamisessa tapahtuu virhe Syötteet ovat virheelliset
Jälkiehdot:	Tallennetaan muutokset tietokantaan

Käyttötapaus: Varauksen poistaminen

Yhteenveto:	Varauksen poistaminen tietokannasta
Toimijat:	Oppilas/Opettaja
Esiehdot:	Oppilas on varannut ajan ja hänelle on lähetetty sähköpostiin peruutuskoodi
Kuvaus:	Etusivulla on lomake johon syötetään peruutuskoodi Tulostetaan ajan tiedot, jotka ovat peruutuskoodin ajalle Varmistetaan poistaminen
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe Tietokantaan tallentamisessa tapahtuu virhe Syötteet ovat virheelliset
Jälkiehdot:	Varaus poistuu ajalta ja se on varattavana taas uudestaan

Käyttötapaus: Varauksen poistaminen

Yhteenveto:	Varauksen poistaminen tietokannasta
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Tulostetaan varaukset ja valitaan oikea varaus Tulostetaan varaajan tiedot Valitaan ”Poista varaus” Varmistetaan poistaminen Poistetaan varaus
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe Poistamisessa tapahtuu virhe
Jälkiehdot:	Varaus poistetaan tietokannasta

Käyttötapaus: Selaaminen/viikkokohtaiset varaukset

Yhteenveto:	Viikkokohtaisten varauksien selaaminen
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Ohjelma listaa viikon varaukset Tulostus sisältää seuraavat tiedot Aloitusaika Lopetusaika Aihe
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe
Jälkiehdot:	Tulostetaan viikon varaukset

Käyttötapaus: Selaaminen/Tietokannan selaaminen eri kriteereiden mukaan

Yhteenveto:	Tietokannan selaaminen eri kriteereiden mukaan
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Tulostetaan kaikki varaukset tai ajat jotka voidaan varata Listaa voidaan järjestää eri kriteereiden mukaan Ajat: Päivän, etunimi tai sukunimen mukaan Varaukset: Päivän, etunimi tai sukunimen mukaan
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe
Jälkiehdot:	Tulostetaan tietokanta

Käyttötapaus: Tietokannan hallinta/Tietokanta xls -tiedostoksi

Yhteenveto:	Tietokannan taulujen tuonti xls –tiedostomuotoon (Excel)
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Valitaan mistä taulusta halutaan xls –tiedosto Tallennetaan tiedosto tietokoneelle
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe
Jälkiehdot:	Tietokannasta saadaan tehtyä xls -tiedosto

Käyttötapaus: Tietokannan hallinta/Tyhjentäminen

Yhteenveto:	Tietokannan taulun tyhjentäminen
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Valitaan kumpi taulu halutan tyhjentää Annetaan salasana Varmistetaan poistaminen Tietokannan taulu tyhjennetään
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe Annettu salasana on virheellinen Poistamisessa tapahtuu virhe
Jälkiehdot:	Taulu tyhjennetään kokonaan

Käyttötapaus: Tietokannan hallinta/varmuuskopiointi/Vienti

Yhteenveto:	Tietokannan taulun tuonti tietokoneelle SQL/CSV –muotoisena tiedostona
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Valitaan kummasta taulusta halutaan varmuuskopio Selain avaa tiedoston latausikkunan
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe
Jälkiehdot:	Varmuuskopio tallentuu opettajan tietokoneelle

Käyttötapaus: Tietokannan hallinta/varmuuskopiointi/Tuonti

Yhteenveto:	Tietokannan taulun tuonti SQL/CSV –muotoisesta tiedostosta
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Valitaan minkä taulun haluaa tuoda tietokoneelta Annetaan salasana Syötetään lomakkeella tiedoston sijainti tietokoneella Hyväksytään tiedoston tuonti
Poikkeukset:	Tiedoston avaamisessa tapahtuu virhe Tiedoston viennissä tapahtuu virhe
Jälkiehdot:	Varmuuskopion tiedot tallentuvat tietokantaan

Käyttötapaus: Tietokannan hallinta/Vanhojen aikojen ja varauksien poistaminen

Yhteenveto:	Kaikkien niiden aikojen ja varauksien poistaminen, jotka ovat vanhempia, kuin tämä päivä
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Varmistetaan poistaminen Poistetaan ajat ja varaukset
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe Poistamisessa tapahtuu virhe
Jälkiehdot:	Vanhat ajat poistuvat tietokannasta

Käyttötapaus: Tietokannan hallinta/Päivien poistaminen tietyltä aikaväliltä

Yhteenveto:	Aikojen poistaminen tietyltä aikaväliltä
Toimijat:	Opettaja
Esiehdot:	Opettaja on kirjautunut hallintaan
Kuvaus:	Tulostetaan lomake, johon syötetään aikaväli Varmistetaan poistaminen Tiedustellaan lähetetäänkö varauksen tekijöille sähköposti Lähetetään viesti: tulostetaan lomake, johon voidaan kirjoittaa otsikko ja viesti ja tämän jälkeen poistetaan aikavälin ajat ja lähetetään varauksen tehneille viesti Ei lähetetä viestiä: Poistetaan ajat ja varaukset tietokannasta
Poikkeukset:	Tietokannan avaamisessa tapahtuu virhe Poistamisessa tapahtuu virhe
Jälkiehdot:	Ajat ja varaukset poistetaan annetulta aikaväliltä

Ei toiminnalliset vaatimukset

Ei-toiminnallisiin vaatimuksiin kuuluu FURPS+ periaatteen mukaisesti käytettävyys, luotettavuus, suorituskyky, tuettavuuteen, sekä muut ei-toiminnalliset ominaisuudet.

Käytettävyys

Oppilaille tarkoitettu ajanvarausjärjestelmä suunnitellaan helppokäyttöiseksi, jotta kaikki voisivat käyttää ohjelmaa ilman, että tarvitsee ensin opetella sen käyttöä.

Luotettavuus

Ohjelman pitää olla luotettava, jotta sillä tehdyt varaukset toimisivat. Myös mahdolliset virhetilanteet on otettava huomioon, jotta ohjelma toimisi silloinkin oikein.

Suorituskyky

Suorituskyky ilmaiseen, miten nopeasti järjestelmä toimii. Järjestelmän nopeuteen vaikuttavat sivujen koko (kuvat nostavat tarvittavan latauksen määrää). Myös tietokannan koko vaikuttaa ohjelman nopeuteen, sillä mitä isompi kanta sitä hitaampi haku.

Tuotettavuus

Tuettavuudella ilmaistaan, miten hyvin järjestelmä toimii erilaisissa käyttöympäristöissä. Järjestelmää käytetään web-selaimella joten ohjelman on toimittava yleisimmillä selaimilla.

Muut ei toiminnalliset vaatimukset

Ohjelmaa tehtäessä on tehtävä helppolukuista koodia (kommentointi, sisennykset jne.) sillä silloin se mahdollistaa ohjelman helpon päivittämisen, kuin myös jatkokehityksen. Projektia suunniteltaessa ei pyritä erikseen optimoimaan suorituskykyä tai tietokannan kokoa.

Rajoitteet

Rajoite	Kuvaus
Ohjelmointikielet	xHTML,PHP,SQL
Tietokannan tallennusmuoto	SQL-pohjainen tietokanta (MySQL)
Käyttöliittymä	Pitää tukea yleisimpiä selaimia
Palvelin	Microsoft IIS 6.0, MySQL (v5.0.22) ja MSSQL ja PHP versio 5.2.5-dev
Ohjeistus	Word/html
Järjestelmän toimitus	Palvelimelle asennettuna
Kieli	Suomi/Englanti

LIITE 2. Tietokantaluokan koodi

```
<?php
class tietokanta
{
		var $palvelin;
		var $kayttaja;
		var $salasana;
		var $tietokanta;

		var $rivit;
		var $tulokset;
		var $yhteys;
		var $valmius;

		public function __construct()
		{
			$this->tulokset = null;
			$this->valmius = false;

		}
		public function __destruct(){ @mysql_close($this->yhteys); }

		public function setPalvelin($palvelin){ $this->palvelin = $palvelin; }
		public function setKayttaja($kayttaja){ $this->kayttaja = $kayttaja; }
		public function setSalasana($salasana){ $this->salasana = $sala-
sana; }
		public function setTietokanta($tietokanta){ $this->tietokanta = $tie-
tokanta; }

		public function ase-
ta_yhteystiedot($palvelin=null,$kayttaja=null,$salasana=null,$tietokanta=null)
		{
			if(!isset($palvelin,$kayttaja,$salasana,$tietokanta))
				die("Tarkista yhteyden tiedot");
			$this->setPalvelin($palvelin);
			$this->setKayttaja($kayttaja);
			$this->setSalasana($salasana);
			$this->setTietokanta($tietokanta);
			$this->isReady = true;
		}
		public function yhdista()
		{
			if(!$this->isReady) die("not ready to connect
".mysql_error($this->yhteys));
```

```
                $this->yhteys = mysql_connect($this->palvelin,$this->kayttaja,$this->salasana) or die("Ei voitu yhdistää tietokantaan ".mysql_error($this->yhteys));
                $this->kanta($this->tietokanta);
                //$this->kysely("SET NAMES 'utf8'",$this->yhteys); //UTF-8
        }

        public function kanta($tietokanta)
        {
                $this->setTietokanta($tietokanta);
                mysql_select_db($this->tietokanta,$this->yhteys) or die("Tietokantaan ei löydy.".mysql_error($this->yhteys));
        }

        public function kysely($sql)
        {
                $this->tulokset = mysql_query($sql,$this->yhteys) or die("Virheellinen kysely.".mysql_error($this->yhteys));
        }

        public function kyselyn_rivimaara($selectMode = true)
        {
                if($selectMode)
                {
                        return mysql_num_rows($this->tulokset);
                }
                else
                {
                        return mysql_affected_rows($this->yhteys);
                }
        }


        public function kyselyn_rivit()
        {
                if(!$this->tulokset) die("Nothing found!".mysql_error($this->yhteys));
                $this->rivit = array();
                while($r = mysql_fetch_array($this->tulokset,MYSQL_ASSOC))
                {
                        $this->rivit[] = $r;
                }
                mysql_free_result($this->tulokset);
                return $this->rivit;
        }
```

```
		public function yhteyden_tarkistus()
		{
			if($this->yhteys)
			{
				return "Yhdistetty";
			}
			else
			{
				return "Ei yhdistetty";
			}
		}
		public function sulje()
		{
			mysql_close($this->yhteys);
			$this->yhteys = null;
		}


}
?>
```