Opinnäytetyö (AMK)

Tietotekniikan koulutusohjelma

Sulautetut järjestelmät

2013

Henry Holmberg

AJANVARAUSJÄRJESTELMÄN JA ASIAKASREKISTERIN SUUNNITTELU JA TOTEUTUS XAMPP-YMPÄRISTÖSSÄ

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikan koulutusohjelma | Sulautettujen järjestelmien ohjelmistotekniikka

Toukokuu 2013 | Sivumäärä: 60

Ohjaaja: TkL Jari-Pekka Paalassalo

Henry Holmberg

AJANVARAUSJÄRJESTELMÄN SUUNNITTELU JA TOTEUTUS XAMPP-YMPÄRISTÖSSÄ

Tässä opinnäytetyössä suunniteltiin ja toteutettiin XAMP-ympäristössä toimiva ajanvarausjärjestelmä sekä tietojen selaamiseen tarkoitettu asiakasrekisteri. Tarkoituksena oli toteuttaa helppokäyttöinen ja selkeä tietokannan kanssa kommunikoiva ajanvarausjärjestelmä osaksi sivustorakennetta. Opinnäytetyön tavoitteena oli tukea liiketoimintaa ja helpottaa informaation hallintaa.

Työn suunnittelua varten kartoitettiin asiakkaan tarpeet, joista johdettiin toteutukseen tarvittavat ominaisuudet. Ajanvarausjärjestelmän keskeiset ominaisuudet ovat interaktiivinen kalenteri, tietokantarakenne sekä automaattinen postitusfunktio. Asiakasrekisteri suunniteltiin tukemaan tätä ratkaisua mahdollistamalla helppo informaation hallinta tietokannan kautta eri listausmahdollisuuksilla ja visuaalisella presentaatiolla. Toteutukseen käytettiin PHP, MySQL ja HTML –kieliä.

Lopputuloksena saatiin luotua käyttäjäystävällinen ja ulospäin yksinkertainen moduuli, joka on helppo liittää osaksi sivustorakennetta ja on muokattavissa erilaisiin tarpeisiin.

ASIASANAT:

XAMPP, Apache, MySQL, PHP, ohjelmointi

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded Software

May 2013 | Total number of pages: 60

Instructor: Jari-Pekka Paalassalo , Lic. Tech, Principal Lecturer

Henry Holmberg

THE DESIGN AND IMPLEMENTATION OF APPOINTMENT SYSTEM AND CUSTOMER REGISTRY IN XAMPP ENVIROMENT

In this thesis an appointment system and customer registry was designed and implemented in an XAMPP environment. The main idea was to implement an easy to use and transparent system that communicates with the database and is a part of the site structure. The goal of the project was to support a business solution and ease the control of customer information.

To support the design part of the project, the client´s customer needs were surveyed and the features needed for the project were derived from that survey. The main features of the appointment system were: interactive calendar, database design and automated mailing function. The appointment system was designed to support these needs by enabling easy information control via database, enabling the use of different search parameters and visual presentation. To create this solution, the PHP, MySQL, and HTML languages were used.

As an end result, a user-friendly and simple module was created. The module is easy to attach as part of a site structure and edit for various needs.

KEYWORDS:

XAMPP, Apache, MySQL, PHP, programming

SISÄLTÖ

KÄYTETTY SANASTO **7**

1 JOHDANTO **9**

2 VALITTUJEN OHJELMOINTIKIELIEN MÄÄRITTELY **10**

3 TIETOTURVA OHJELMISTOISSA **13**
3.1 Injektiot 13
3.2 Luottamuksellisuus ja pääsynvalvonta 13
3.3 Eheyden valvonta relaatiotietokannassa 14

4 TESTAUS OSANA OHJELMISTOKEHITYSTÄ **15**
4.1 Testausmenetemät osana ohjelmistokehitystä 15
4.2 Testausmenetelmät: Valkolaatikko 16
4.3 Testausmenetelmät: Mustalaatikko 18
4.4 Testausmenetelmät: Harmaalaatikko 20

5 ASIAKASVAATIMUKSET **21**

6 TOTEUTUS **24**
6.1 Kalenteri 24
6.2 Tietokanta 25
6.3 Asiakasrekisteri 26

7 OHJELMISTORAKENTEEN KUVAUS SEKÄ KAAVIOT **27**
7.1 Ajanvarausjärjestelmän ja asiakasrekisterin dokumentaatio 27
7.2 Tiedostokuvaus: Kalenteri 27
7.3 Tiedostokuvaus: Ajanvarausjärjestelmä 35
7.4 Tietokantakuvaus 37
7.5 Postitustoimintojen kuvaus 39

8 TIETOTURVA **41**
8.1 Käyttäjäsyötteiden sanitointi 42
8.2 Tietokantaturvallisuus 43

9 OHJELMISTOTESTAUS **45**

9.1 Testaus 45
9.2 Ajanvarauksen testitapaukset 46
9.2.1 Testitapaus 1: kalenterin luotettavuus 46
9.2.2 Testitapaus 2: kalenterin linkit 47
9.2.3 Testitapaus 3: palveluntilaaja-lomake 48
9.2.4 Testitapaus 4: kontaktihenkilö-lomake 49
9.2.5 Testitapaus 5: tapahtumatiedot-lomake 50
9.2.6 Testitapaus 6: postitusfunktioiden toiminnallisuus 51
9.3 Asiakasrekisterin testitapaukset 52
9.3.1 Testitapaus 1: login-lomakkeen tietojen sanitointi 52
9.3.2 Testitapaus 2: asikastietojen listaus 52
9.3.3 Testitapaus 3: asiakastietojen poistaminen 53
9.3.4 Testitapaus 4: uloskirjautuminen 54

10 YHTEENVETO 56

LÄHTEET 58

LIITTEET

Liite 1. Tietokannan relaatiomalli ja tietuekuvaukset.
Liite 2. Ohjelman prosessikaavio: Ajanvarausjärjestelmä.
Liite 3. Ohjelman prosessikaavio: Asiakasrekisteri.

KUVAT

Kuva 1. Esimerkki ohjelmistorakenteesta. [12] 16
Kuva 2. Esimerkki kaikista mahdollisista ohjelmiston poluista. [12] 17
Kuva 3. Esimerkki ehtolauseiden poluista. [12] 17
Kuva 4. Esimerkki mustalaatikkotestauksen ideasta. 18
Kuva 5. Vanha toimintamalli. 22
Kuva 6. Rinnakkaisen uuden toimintamallin tuoma etu. 23
Kuva 7. Yrityksen uusi toimintamalli erityistarpeita vaavissa tilauksissa. 23
Kuva 8. tilaus.php:n päätoiminnot. 27
Kuva 9. kielivalikko.php 28
Kuva 10. kalenteri.php ja varatut_päivät.php muuttujineen ja toimintoineen. 29
Kuva 11. päiväys_käsittelijä.php muuttujineen ja toimintoineen. 30
Kuva 12. tietokanta_varaus_tarkistus.php muuttujineen ja toimintoineen. 30

Kuva 13. palvelu_tilaaja_lomake.php muuttujineen ja toimintoineen. 30
Kuva 14. kontakti_lomake.php muuttujineen ja toimintoineen. 31
Kuva 15. kontakti_turva.php toimintoineen. 31
Kuva 16. palvelu_valinta_lomake.php muuttujineen ja toimintoineen. 32
Kuva 17. palvelu_.php toimintoineen. 33
Kuva 18. Mysql_kirjoitus.php toimintoineen. 33
Kuva 19. *postitin* 1, 2 .php ja palvelu_tiedostot-sposti.phpn relaatiot ja toiminnot. 34
Kuva 20. tietokanta_yhteys.php muuttujineen ja toimintoineen. 34
Kuva 21. Sisaankirjaudu.php muuttujineen ja toimintoineen. 35
Kuva 22. Tietokanta_tarkistus.php muuttujineen ja toimintoineen. 35
Kuva 23. Asiakastieto_lomake.php muuttujineen ja toimintoineen. 36
Kuva 24. Listari muuttujineen ja toimintoineen. 36
Kuva 25. Tietokanta_haut.php muuttujineen ja toimintoineen. 37
Kuva 26. Esimerkki PHP mail funktion parametreista ja käytöstä. 40
Kuvio 30. Esimerkki filtterien käytöstä. 43

KÄYTETTY SANASTO

Apache http-palvelin	Avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma
BOOLEAN	Tietokannassa tinyint-tyyppinen tietue, joka voi saada arvon 1 tai 0
DATE	Tietokannoissa päiväystyyppi, joka annetaan Yhdysvaltalaisessa muodossa (VVVV-KK-PP)
DNS	Nimipalvelujärjestelmä, joka kääntää annetut nimet ip-osoitteiksi (Domain Name Service)
ENUM	Enumeration (tietokannoissa) tietue voi saada vain ennalta määriteltyjä arvoja
FiSTB	Suomalainen ohjelmistotestaamisen lautakunta (Finnish Software Testing Board)
GPL	Avoimen lähdekoodin levityslisenssi (General Public License)
Hash-funktio	Mikä tahansa sellainen kuvaus, joka kuvaa mahdollisten avainten joukon K_Set kokonaislukujen (0,..., H_Size-1) joukolle (salaustapa)
HTTP	Protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon
MySQL	Relaatiotietokantaohjelmisto (My Structures Query Language)
Perl	Ohjelmointikieli
PHP	Ohjelmointikieli
Primääriavain	Uniikkiavain tietokannan tietueissa, jolla luodaan uusia relaatiota ja tunnistetaan, mitkä tiedot kuuluvat yhteen
Sateenkaaritaulu	Nopea tapa selvitää henkilöiden salasanat tunnetuista salausjärjestelmistä
SMTP	TCP-protokolla, jota käytetään viestien välittämiseen säköpostipalvelimien kesken (Simple Mail Transfer Protocol)
Suolaus	Tapa lisätä salattaviin tietoihin alfanumeerinen kirjainyhdistelmä tiedon alkuun ja loppuun
VARCHAR	Variable character (kirjaintyypin muuttuja)
Viiteavain	Tietokannan väline, joka liittää taulujen väliset tiedot yhteen relaatiolla ja säilyttää viite-eheyttä

XAMPP	Käyttöjärjestelmäriippumaton ohjelmistopaketti

1 JOHDANTO

Nykypäivänä ihmiset pystyvät tilaamaan palveluita ja tuotteita verkosta helposti sekä nopeasti. Tämänkaltainen trendi on aiheuttanut yrittäjille tarvetta miettiä myös omaa liiketoimintamallia ja ottamaan huomioon asiakkaiden tarpeen omilla sivustoillaan.

Internetin kautta tapahtuvaan palveluiden tilaamiseen yritykset hyödyntävät omia verkkosivujaan. Pienikään yritys ei voi jättää huomioimatta ihmisten lisääntynyttä tapaa tilata tarvitsemansa tuotteet ja palvelut verkosta, joten nousee tarve osata kartoittaa tarvitsemansa verkkopalvelun laajuus ja toteuttamistapa. Tämän työn tarkoitus oli vastata toimeksiantajan tarpeisiin valokuvauspalveluita tuottavan yrityksen kannalta. Yrityksen asiakastarpeiden kannalta riittäväksi tavaksi muotoutui ajanvarausjärjestelmän tuottaminen osaksi yrityksen verkkosivuja. Järjestelmä pystyy itsenäisesti vastaamaan ihmisten ajanvarauksiin ja näin vapauttaa enemmän aikaa yritykseltä palveluiden toimittamiseen, sen sijaan että yritys joutuisi vastaamaan jatkuvasti varauksiin.

Opinnäytetyössä toteutettiin yritykselle kalenteri-, postitus- ja asiakasrekisterimoduulit, jotka pystyttäisiin liittämään osaksi yrityksen verkkosivuja. Opinnäytetyössä käsitellään asiakkaan tarpeiden kartoittamisen pohjalle rakentuvaa moduulien suunnittelua, toteutusta sekä testausta. Näin pyrkien kattamaan koko ohjelmistosuunnittelun pääkohdat noudattaen toimivia kehitysmalleja.

2 VALITTUJEN OHJELMOINTIKIELIEN MÄÄRITTELY

Ohjelmointikieliä valittaessa pyrittiin tekemään valinnat ajatellen niin yrityksen tarpeita kuin ohjelmoijan osaamista. Lisäksi tärkeänä osana oli valittavien ohjelmointikielien hyvä dokumentaatio, jotta tarvittaessa ohjelmistoa pystyisi jatkamaan, muokkaamaan ja ylläpitämään kolmannen osapuolen tahot. XAMPP-ratkaisu valittiin toteukseen sen helppokäyttöisyyden sekä sen käyttämien ohjelmointikielten vuoksi. Kaikki XAMPPin sisältämät kielet ovat jo olleet markkinoilla pitkään ja ne ovat saavuttaneet laajan suosion. Lisäksi ne ovat hyvin dokumentoituja niin käyttäjien kuin ne tuottaneiden yritystenkin puolesta. Näiden lisäksi toteutusmuoto tukee palvelintilan valintaa asiakkaan puolesta. Asiakkaan vuokraamalla palvelimella on samat ympäristöt käytössä kuin XAMPP-ratkaisussa, sillä erolla että Apachen ja PHPn konfiguraatio määrittelyt ovat palvelimen omistavan tahon muokkaamia ja näin ollen tietoturva ratkaisut eivät jää näiltä osin ohjelmoijan vastuulle, vaan siitä vastaa palvelimen vuokraava yritys.

Apache HTTP Palvelinohjelmisto on avoimeen lähdekoodiin perustuva ohjelmisto. Apache ohjelmistosäätiö on 1999 perustettu voittoa tavoittelematon säätiö, jonka tarkoitus on tuottaa, tutkia ja kehittää Apache-lisenssin alaisia ohjelmia, joilla on GPLv3 yhteensopiva lisenssi. Tässä toteutuksessa kyseinen lisenssi tarkoitti, ettei Apache HTTP Palvelinohjelmistosta tulisi minkäänlaisia kustannuksia yritykselle. Tämän vuoksi se valittiin käytettäväksi ohjelmiston toteutuksessa. [1]

MySQL on myös GPL-lisenssin alainen ohjelmisto, joskin GPL-lisenssi ei päde tapauksissa, joissa ohjelmistoa alettaisiin myydä tuotteena eteenpäin muille yrityksille. MySQL on maailman suosituin avoimen lähdekoodin tietokantaohjelmisto, joskin on hieman kyseenalaista luokitellaanko sitä enää avoimeen lähdekoodiin johtuen Oraclen luomasta vaikeaselkoisesta maksullisen lisenssin ja ilmaisen lisenssin käyttöoikeusmäärittelyistä. Tietokantana toimii MySQL-versio 5.5.25a, joka on GPL–lisenssin alainen. [2]

Tietokannan moottoriksi valittiin InnoDB, koska testauksen mukaan se suoriutuu huomattavasti paremmin melkein kaikilla osa-alueilla kuin MyIsam. [3]

PHP on niin sanottu komentosarjakieli. Koska PHP suoritetaan palvelimella, sen etuna on, ettei käyttäjiä tarvitse erikseen kehottaa varmentamaan, että heidän selaimessaan on käytössä jokin tietty tuki, jota ohjelma vaatisi (esim. JavaScript). Koska PHP suoritetaan vasta palvelimella aiheuttaa se ”ylimääräistä” kuormitusta palvelimella. Keskisuurilla tai suurilla yrityksillä, joiden sivustoja käyttävän monet tuhannet ihmiset, täytyy PHP:n aiheuttama palvelinkuormitus ottaa huomioon. Kuormituksesta ei tarvitse tässä kyseisessä projektissa olla huolissaan, sillä tilaajayrityksellä on pieni henkilöstö ja se toimii matalalla volyymillä. Tilaajayritys ei pyri tai pystyisikään palvelemaan suuria ihmismääriä. Yritys pyrkii lähinnä käyttämään Internet-palvelua saadakseen uusia asiakaskontakteja ja pystyäkseen kasvattamaan toimintaansa. Toiminnan arvioitu kasvu ja yrityksen rajalliset resurssit huomioiden voidaankin turvallisesti olettaa, ettei palvelimelle tule liiallista rasitusta. Tämän lisäksi ohjelmoijalla itsellään oli eniten kokemusta PHP:stä, joten kyseisen kielen valitsemisella pystyttiin takaamaan parempi ohjelmoinnillinen vakaus, turvallisuus ja olisi mahdollista edetä nopeammin. Näin pystyttiin vastaamaan asiakkaan vaateisiin kustannustehokkuudesta ja toimivuudesta. Perl-kieltä ei käytetty, koska PHP valittiin esitettyjen vahvuuksiensa takia. [4]

Erilaisia ohjelmistokehityskehikkoja (framework) ei myöskään käytetty. Syynä oli ohjelmoijan huono tuntemus PHP:n ohjelmistokehityskehikkoista ja näihin tutustuminen olisi vienyt huomattavan määrän aikaa. Lisäksi ohjelmistokehityskehikkoja käytettäessä on mahdollista joutua selvittämään ennalta tuttujen toimintojen toteuttamista ohjelmistokehityskehikkojen vaatimilla ehdoilla. Tämä olisi tarkoittanut opinnäytetyön laajentumista käsittelemään myös eri ohjelmistokehityskehikkoja ja se ei ollut aikataulullisesti mahdollista. Tällöin olisi pitänyt määrittää mm. mitä versioita PHP:stä ja MySQL:stä mikäkin ohjelmistokehityskehikko tukee. Lisäksi olisi pitänyt varmistaa että valittu kehikko varmasti on yhteensopiva tietokannan sekä palvelimen kanssa. Näistä asioista aiheutuu myös erilaisia potentiaalisia turvallisuusuhkia. Lisäksi

ohjelmistokehityskehikkojen kirjastoissa on potentiaalinen mahdollisuus tietoturvauhkien periytymiseen. Tarkemmin tietoa eri ohjelmistokehityskehikkojen haavoittuvuuksista löytyykin internet-hauilla, joilla useat ohjelmistokehityskehikkojen tarjoajat ovat antaneet sivuilleen paketteja eri haavoittuvuuksien tukkimiseksi. Lisäksi ohjelmistokehityskehikkojen tarjonnasta olisi ollut vaativaa löytää omiin tarpeisiin sopiva ratkaisu, ilman kokemusta tai laajaa ennakkokäsitystä. [5]

3 TIETOTURVA OHJELMISTOISSA

Tietoturva on tärkeä osa ohjelmistojen toteutusta ja se tulisi pystyä huomioimaan erityisesti verkoissa toimivissa järjestelmissä. Tässä kappaleessa tarkastellaan tietoturvan eri aspekteja.

3.1 Injektiot

Injektioilla tarkoitetaan koodin syöttämistä käyttäjille varattujen kirjoituskenttien kautta. Injektiot ovat vaara jokaisella käytetyllä ohjelmointikielellä, jossa otetaan vastaan syötteitä käyttäjiltä, eivätkä ne ole minkään tietyn kielen vitsauksia. Injektoidulla koodilla yritetään vaikuttaa ohjelman kulkuun. Paras tapa varautua injektioita vastaan onkin kohdella käyttäjän syötteitä vihamielisenä (hostile) ja aina valitoida ja suodattaa käyttäjän syötteet hänen annettuaan ne. Lisäksi ohjelman tietoturvan tarpeen mukaan voidaan jopa koodata kaikki käyttäjän syötteet sekä tulosteet toiseen muotoon, jolloin injektioista tulee lähes mahdottomia suorittaa. [6]

3.2 Luottamuksellisuus ja pääsynvalvonta

Luottamuksellisuudella pyritään varmistamaan, ettei annettuja tietoja pääse katselemaan tai muokkaamaan muut kuin ne, joilla on oikeudet tähän. Luottamuksellisuutta voidaan toteuttaa pääsynvalvonnalla jolloin henkilöille, joilla on oikeus katsoa tai muokata tietoa, annetaan yksilöllinen tunnistautumiskeino. Tunnistautumiskeino voi käsittää kaikkea salasanasta avainkorttiin tai tietyiltä tietokoneilta suoritettavaan sisällön hallintaan. [7]

3.3 Eheyden valvonta relaatiotietokannassa

Tiedon eheys termi tulee usein esiin varsinkin tietokannoista lukiessa. Eheydellä tarkoitetaan sitä, että tieto on tarkkaa ja yhdenmukaista sen syöttämishetkestä sen poistamiseen asti. Tiedon eheyttä pyritään valvomaan yleensä normaalimuodoilla. Normaalimuotoja on 9 erilaista (1NF, 2NF, 3NF, EKNF, BCNF, 4NF, 5NF, DKNF, 6NF), mutta tässä käsitellään vain neljää yleisintä niistä, eikä lainkaan niin sanottua vahvaa kolmatta normaalimuotoa (BCNF). [8]

Ensimmäinen normaalimuoto vaatiin, että jokaisen tietokannan taulun sarakkeiden tulee olla atomisia. [9]

Toisessa normaalimuodossa ensimmäisen normaalimuodon kriteerit tulee olla täytetty. Tämän lisäksi kaikkien sarakkeiden pitää olla riippuvaisia koko avaimesta. Avaimella tarkoitetaan sitä, että relaation ominaisuuden tulee olla täydellisesti riippuva jokaisesta relaation avainehdokkaasta. [9]

Kolmannessa normaalimuodossa pitää olla toisen normaalimuodon kriteerit täytetty. Tämän lisäksi kaikkien sarakkeiden tulee olla riippuvaisia pääavaimesta. Pääavain on valittu avainehdokkaista, joten kaikkien muiden sarakkeiden tulee olla riippuvaisia siitä. [9]

Neljännessä normaalimuodossa pitää olla kolmannen normaalimuodon kriteerit täytetty. Tämän lisäksi taulujen sarakkeissa ei saa esiintyä kahden tai useamman ominaisuuden moniarvoista riippuvuutta, jos nämä ominaisuudet ovat keskenään riippumattomia. [10]

4 TESTAUS OSANA OHJELMISTOKEHITYSTÄ

Testaus on ehkä tärkein askel ohjelmistokehityksessä. Hyvin suunnitellun testausprosessin ja dokumentaation pohjalta voidaan havaita ja vaikuttaa moniin ohjelmiston ongelmiin sekä kehittämistarpeisiin ennen kuin tuote on edes päässyt loppukäyttäjälle asti. Tätä kautta taataan asiakastyytyväisyys ja pidetään yllä korkean tason kuvaa yrityksestä tai ohjelmoijasta.

Ammattimaista testausta varten olisi hyvä, jos yrityksellä olisi sertifiointijärjestelmä apunaan. Hyviä tapoja aloittaa tutustuminen testauksen sertifiointijärjestelmiin on tutustua FiSTB:n sivuilla ammattitestaamisen koulutus- ja kumppanuusohjelmiin. FiSTB on suomalainen ohjelmistotestauksen lautakunta, joka jakaa tietoa eri tasojen koulutusohjelmista ja yrityksistä, jotka kurssittavat testaajia sekä järjestävät sertifikaattikokeita. Kuitenkaan kaikissa projekteissa tai yrityksissä ei ole mahdollista hyödyntää ammattimaista testaamista resurssien puutteiden vuoksi. Tällöin olisikin tarpeen suunnitella mahdollisimman kattava sekä tarkka dokumentaatio valituista testausmenetelmistä. [11]

4.1 Testausmenetemät osana ohjelmistokehitystä

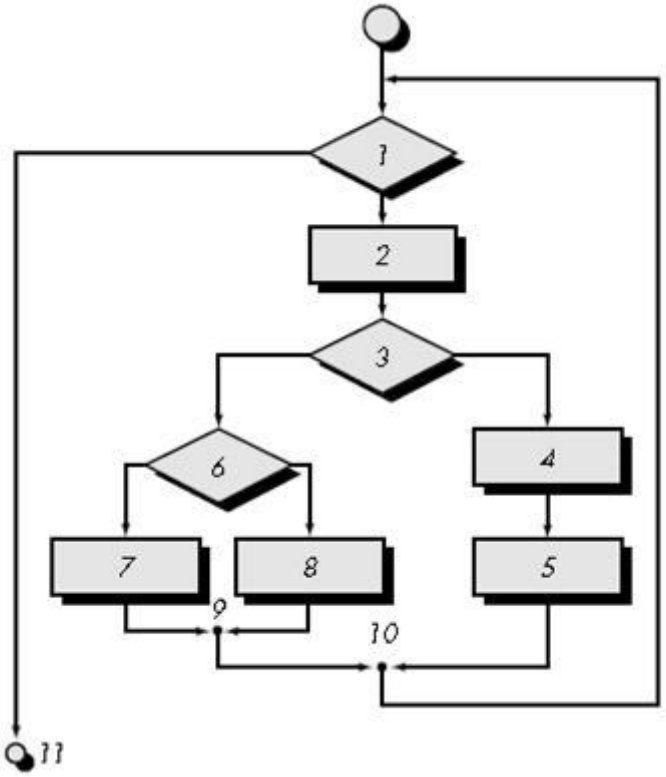
Testauspainotteisia menetelmiä on käytössä useita. Koska menetelmiä on niin yrityksensisäisiä kuin maailmanlaajuisestikin tunnettuja, keskitytään vain niihin, joita on tietyin rajoittein hyödynnetty opinnäytetyössä.

- Valkolaatikko (White-Box)
- Mustalaatikko (Black-Box)
- Harmaalaatikko (Grey-Box).

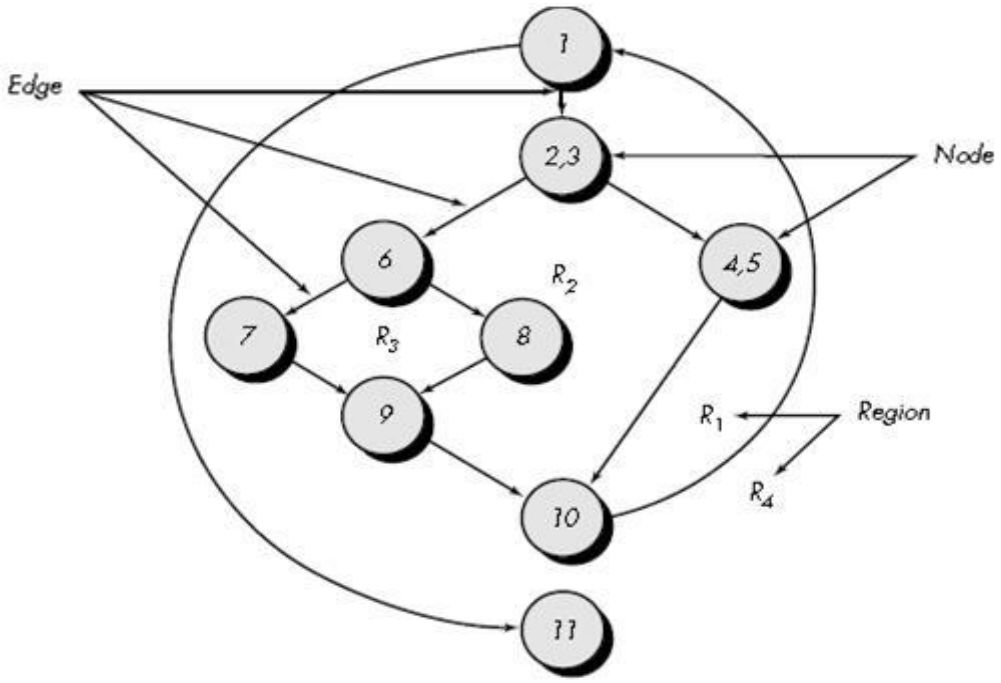
4.2 Testausmenetelmät: Valkolaatikko

Kuten yleensäkin testauksessa, on valkolaatikkotestauksessa tarkoitus löytää mahdollisimman paljon virheitä mahdollisimman vähällä ajankäytöllä ja vaivalla. Valkolaatikkotestaus on käytäntö, jolla testataan ohjelman sisäisiä rakenteita tai aplikaation toimintoja. Valkolaatikkotestaus pyrkii tarkastelemaan asioita ohjelman sisäisestä näkökulmasta. Testaajat pyrkivät valitsemaan ohjelman osioille arvot ja päättelemään ja ennustamaan, mitä kyseisen osion tulisi palauttaa (kuva 1). Yleensä testauksen osioiden pitää noudattaa seuraavia periaatteita:

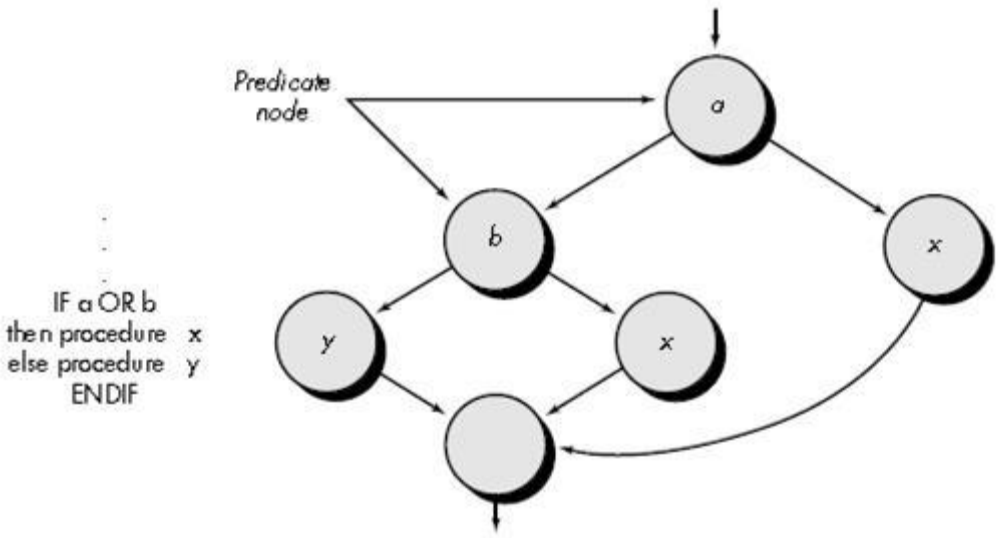
- Käydä läpi kaikki mahdolliset ohjelmapolut (kuva 2)
- Testata kaikki ohjelman saamat totuusarvot, kaikissa ohjelman kohdissa
- Toteuttaa kaikki silmukkarakenteet niiden raja-arvojen sisällä
- Testata kaikki ohjelman sisäiset tiedostorakenteet niiden validoimiseksi (kuva 3). [12]



Kuva 1. Esimerkki ohjelmistorakenteesta. [12]



Kuva 2. Esimerkki kaikista mahdollisista ohjelmiston poluista. [12]



Kuva 3. Esimerkki ehtolauseiden poluista. [12]

Kuvien 1–3 esimerkkiä käyttäen suunnitellaan jokaiselle kohdalle ja poluille (kuvat 2 ja 3) omat testinsä, joiden pitäisi tuottaa ennalta ”ennustettu” tulos

annettujen parametrien mukaan. Testiä suunniteltaessa on huomioitava, pyritäänkö käsittelemään tilanteita, joissa parametreina ohjelmalle syötetään sallittuja arvoja vai sallittujen arvojen alueen ulkopuolisia arvoja. [12]

Lisäksi olettaen että ohjelma tarvitsee tunkeutumistestaamista, toteutetaan se niistä lähtökohdista, että hyökkääjä tuntee koko järjestelmän ja pystyy näin hyödyntämään mahdollisia tietosuoja-aukkoja. [12]

4.3 Testausmenetelmät: Mustalaatikko

Mustalaatikkotestaus keskittyy ohjelman toimivuuteen. Erona valkolaatikkotestaukseen on se, ettei ohjelman sisäisiin funktioihin, polkuihin tai moduuleihin kiinnitetä huomiota, vaan päämääränä on selvittää testauksella, toimiiko ohjelma oikein. Testaajina ei pidä käyttää ohjelman suunnittelijoita, ohjelmoijia tai kolmansia osapuolia, joilla on tietoa siitä, miten ohjelma toimii ja ”miksi” se toimii. Yksinkertaistettuna tarkoituksena on siis eräänlainen loppukäyttäjätestaaminen, jossa käyttäjän syötteiden mukaan ohjelman antama ulostulo ja käyttäytyminen pitäisi pystyä ennustamaan. Mustalaatikkotestauksia on kahta erilaista [13]:

- Ilman käyttäjää suoritettava testaus (laboratoriotestaus)
- Käyttäjän kanssa suoritettava testaus (kenttätestaus).

Kuva 4. Esimerkki mustalaatikkotestauksen ideasta.

Laboratoriotestauksessa, eli ilman käyttäjää suoritettavassa testauksessa, on päämääränä selvittää toimiiko ohjelma odotetulla tavalla, eli mitkä ovat ohjelmalle määritetyt vaatimukset. On olemassa erilaisia lähestymistapoja testaamiseen. Ensimmäinen on testata jokainen ohjelman osa-alue tai funktio erillisenä sekvenssinä. Toinen on testata jokainen moduuli erikseen ohjelman etenemisjärjestyksessä [13].

Laboratoriotestaus jakautuu vielä eri osa-alueisiin, joita ovat

- Volyymitestaus
- Stressitestaus
- Palautustestaus
- Tehokkuustestaus.

Volyymitestauksella pyritään toteamaan, kuinka ohjelma käyttäytyy suurien tietosyötteiden kanssa. Testauksesta johdetulla tiedolla pystytään määrittämään ohjelman tehokkuutta ja löytämään mahdollisia puskureihin tai ohjelmamuistiin liittyviä virheitä tai rajoitteellisuuksia [13].

Stressitestauksessa järjestelmän täytyy käsitellä suuria määriä tietoa sekä useita funktiokutsuja lyhyessä ajassa. Tarkoituksena testauksella on löytää tieto siitä, mitkä ovat ohjelman rajat. Rajoilla tarkoitetaan miten paljon tietoa/funktiokutsuja voidaan ajaa x määrässä aikaa siten, että ohjelma ei kaadu tai lamaannu [13].

Palautustestauksessa järjestelmä kaadetaan, eli aiheutetaan tahallinen kriittinen virhe järjestelmään, jonka seurauksena järjestelmää yritetään palauttaa. Näin saada tietoa siitä, kuinka vaikeaa/helppoa järjestelmä on palauttaa, kuinka paljon tietoa saadaan palautettua (vain tiettyjä osia, kaikki vai tietyltä ajalta tallennetut tiedot) ja mitä askelia joudutaan ottamaan, jotta tiedot saadaan palautettua.

Tehokkuustestaus on aina riippuvainen käytettävän laitteiston suorityskyvystä, ja siksi onkin tarpeen tietää laitteiston suorituskyky niin paperilla kuin myös joidenkin kolmannen osapuolen testien kautta. Ohjelmiston tehokkuudesta on vaikea saada realistista kokonaiskuvaa, koska käyttäjien laitteistot ovat niin

vaihtelevia. Tehokkuustestauksesta saa usein realistisemman kuvan vasta kenttätestauksen kautta, jolloin ohjelmaa ajetaan ulkopuolisten käyttäjien toimesta heidän omilla laitteillaan. Tämän vuoksi onkin hyvä pitää laboratoriotestausta vain suuntaa antavana ja mahdollisena minimi/maksimi vaatimusten määrittelemisen tapana. Realistinen kuva ohjelman tehokkuudesta tulee vasta kenttätestauksen kautta [13].

Kenttätestauksessa pyritään puolestaan seuraamaan testikäyttäjiä heidän normaaleissa työtilanteissaan ja olosuhteissaan. Siinä pyritään selvittämään pääosin käyttökokemuksia ja käytön mukavuutta. Tällä tavoin voidaan selvittää, onko ohjelma tarpeeksi helppokäyttöinen, ovatko valikot selkeitä, saadaanko tulokset tarpeeksi nopeasti, onko ohjelman opasteita riittävästi, ja yleisesti miten ohjelma palvelee käyttäjäänsä. Näin voidaan pyrkiä luomaan kilpailuetua muihin ohjelmiin, jotka toimivat samoilla alueilla ja voidaan selvittää, vastaako tuote odotuksia ja vaatimuksia. Lisäksi kenttätestaus antaa tietoa siitä, miten ohjelma asettuu osaksi testaajien toimintamalleja esimerkiksi yrityksissä, ja miten se integroidaan ja toimii osana käytössä olevia prosesseja ja järjestelmiä [13].

4.4 Testausmenetelmät: Harmaalaatikko

Harmaalaatikkotestaus on yhdistelmä valko- ja mustalaatikkotestausta. Harmaalaatikkotestauksessa testaajalla on ohjelman toiminnoista, funktioista ja algoritmeista ymmärrys, mutta hän ei kuitenkaan tunne koko järjestelmää eikä kaikkia järjestelmän toiminnallisuuksia. Harmaalaatikkotestaus suoritetaan kuitenkin käyttäjän (kenttätestaus) näkökulmasta, eikä lähdekoodi ole testaajan saatavilla [14].

5 ASIAKASVAATIMUKSET

Ensimmäisenä osana opinnäytetyötä oli tärkeää selvittää asiakasyrityksen tarpeet ja vaatimukset toteutuksen suunnittelua varten. Yrityksen päätarpeena oli mahdollisuus tarjota palveluitaan verkon kautta, ilman että asiakas joutuisi ottamaan yritykseen yhteyttä ja odottamaan vastausta.

Yritys on vanhalla toimintamallillaan joutunut käyttämään suuren määrän aikaa tilausten käsittelyyn riippumatta siitä, toteutuuko tilaus (kuva 5). Vanhassa mallissa asiakas on soittanut yritykseen, jolloin asiakkaan kanssa on sovittu alustava suullinen sopimus. Tämän jälkeen on yleensä pitänyt olla vähintään vielä kerran yhteydessä uudestaan asiakkaaseen, jotta voidaan käydä tarkemmin läpi tarvitut palvelut, sekä kirjoittaa sopimus. Toimintamalli on vienyt suuren määrän aikaa yrityksen toiminnalta, ennen kuin on ollut edes mahdollista päästä tilausten toteuttamisasteelle.

Yritykselle olikin tarkoitus päästä pois vanhasta toimintamallista siirtymällä enemmän internetin kautta tarjottaviin palveluihin (kuva 6). Tällä tavoin yritys pystyisi irrottamaan asiakaspalvelussa käytettyjä resursseja itse palveluiden tuottamiseen. Yrityksen näkökulmasta oli tarpeellista pystyä luomaan palveluille oma varausohjelmisto, mistä asiakkaat näkisivät ilman yhteydenottoa, mitkä päivät ovat jo varattuja muiden tilausten tuottamiseen ja mitkä.

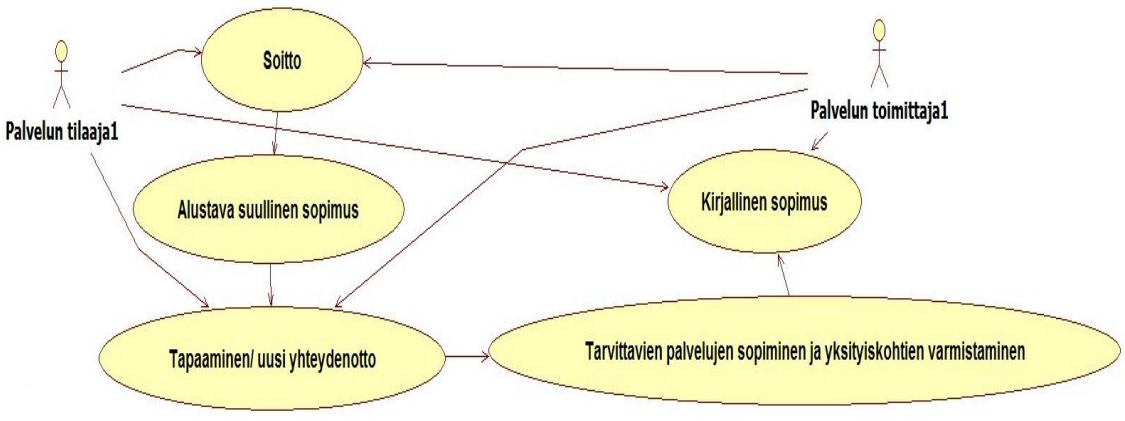
Lisäksi yritys halusi internetin kautta tapahtuvassa asioinnissa tulevan ilmi mitä asiakas tarvitsisi, ottaen huomioon asiakassegmenttien (yhtyeet, yritykset) erikoistarpeet, jotta mahdollisessa neuvottelu tai tilausten tarkentamisvaiheessa olisi osattu jo alustavasti varautua asiakkaan tarpeisiin (kuva 7).

Tällä tavoin pystyttäisiin rajaamaan erikoistarpeita vaativiin asiakasneuvotteluihin vähemmän aikaa, sekä saamaan parempi yleiskuva asiakkaan tarpeista, kun niistä olisi jo alustava tieto ennen ensimmäistä yhteydenottoa. Niiden asiakkaiden, joiden tarpeet eivät vaatineet erityishuomiota, olisi mahdollista varata palvelunsa suoraan internetistä.

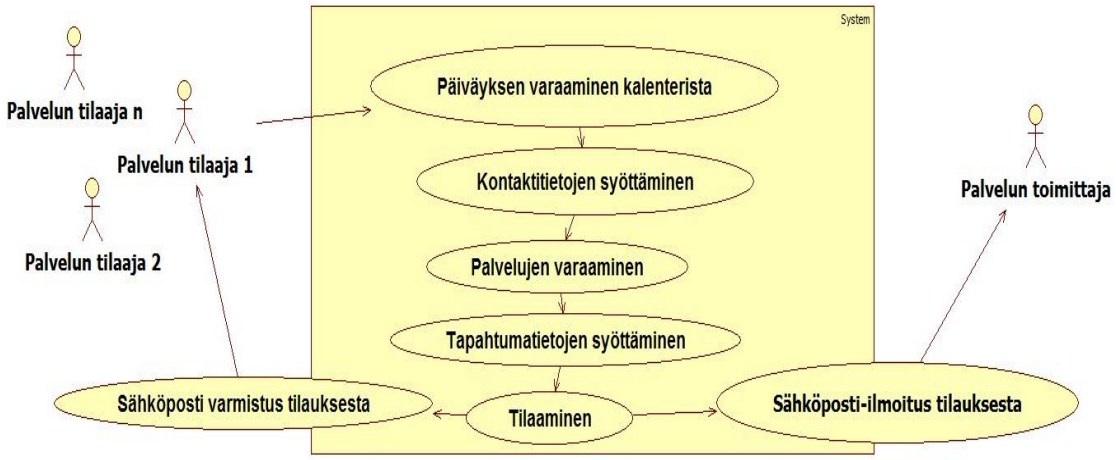
Käyttöliittymän puolesta yritykselle toi lisäarvoa mahdollisuus palvelun tilaamiseen nopeasti ja helposti. Muut tarpeet olivat helppokäyttöisyys niin asiakkaalle kuin yritykselleKin. Toisena asiana opinnäytetyössä tuli yrityksen mahdollisuus pitää asiakkaistaan helpompaa kirjanpitoa, jota voisi käyttää myös tilausten toteutuspäivien tarkasteluun.

Yritys siis tarvitsi helpon tavan tietää, mitkä tilaukset olisi toteutettava minäkin päivänä. Tärkeää oli myös mahdollisuus tarkastella yrityksen verkon kautta tilanneiden asiakkaiden tietoja sekä käyttöliittymän selkeys ja intuitiivisuus. Toteutuksen kannalta oli myös tärkeää, että opinnäytetyön toteutusosa tulisi mahdollisimman kustannustehokkaaksi, koska yritys oli nuori eikä ollut vielä saavuttanut laajaa asiakaskuntaa.

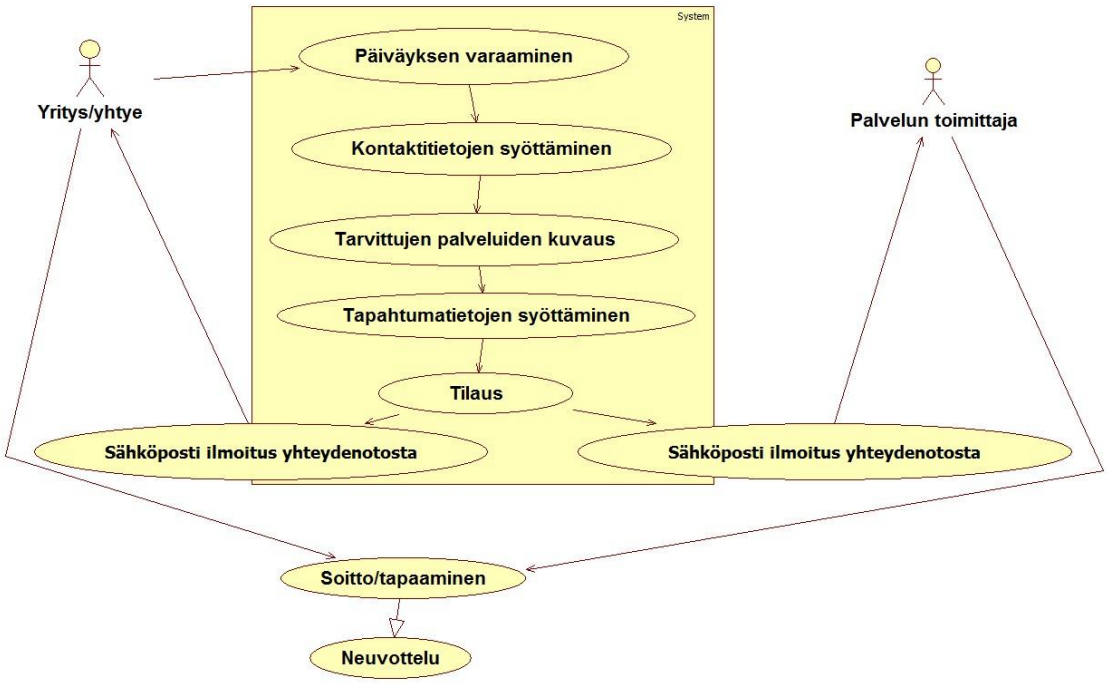
Yrityksen tarpeiden pohjalta hahmoteltiin ajanvarausjärjestemä, joka olisi helppo liittää osaksi verkkosivuja, helppo hallinnoida, sekä selkeä ja käyttäjäystävällinen niin yritykselle kuin palveluiden tilaajillekin. Lisäksi otettiin huomioon mahdollisuus käyttää GPL-lisenssin tai vastaavan alaisia ohjelmia osana ajanvarausjärjestelmää. Tällä tavoitettaisiin kustannustehokkuuden määritelmät sekä nopeutettaisiin projektin etenemistä. Vastaavasti lisääntyneen informaation määrän vuoksi suunniteltiin työn määrää vähentämään asiakasrekisteri, jolla yritys saisi tarpeidensa mukaan listattua eri näkymiin asiakkaiden tietoja ja poistettua virheellisiä tietoja tai toteutumattomia tilauksia.



Kuva 5. Vanha toimintamalli. Asiakas ja palvelun toimittaja joutuvat olemaan kontaktissa kaksi tai kolme kertaa yhtä tilausta kohden.



Kuva 6. Rinnakkaisen uuden toimintamallin tuoma etu. Asiakkaat pystyvät jättämään tilauksensa ja palvelun toimittaja pystyy käsittelemään ne sähköpostin välityksellä.



Kuva 7. Yrityksen uusi toimintamalli erityistarpeita vaavissa tilauksissa. Yrityksellä on jo ennen neuvotteluja alustavat tiedot tarpeista, päiväyksistä ja tapahtumapaikoista.

6 TOTEUTUS

Toteuttaminen aloitettiin käymällä läpi asiakkaan tarpeet ja miettimällä minkälainen runko ohjelmistolle tulisi luoda sekä minkälaisia funktioita ja moduuleita mahdollisesti tarvittaisiin. Luonnollisesti ohjelma muuttui jonkin verran sen mukaan, kun tarpeet muuttuivat joko valokuvaamon puolelta, tai esille tuli jokin seikka, joka sopisi toteutukseen paremmin tai toisi mahdollista lisäarvoa.

6.1 Kalenteri

Ensimmäisenä toteutettavana asiana oli kalenteri, koska se oli ohjelman toiminnallisuuden, asiakkaan sekä kokonaisuuden kannalta tärkein. Tietokanta oli luonnollisesti myös tärkeä, mutta kalenteri oli tehokkaampaa luoda ensin. Kun se olisi valmis, olisi sitä helppoa käyttää tietokannan, sekä muiden moduulien toiminnallisuuden testaamiseen.

Ensimmäiseksi aloitettiin tutkimalla Internetistä, olisiko mahdollisesti tarjolla valmiita kalentereita, joista jonkin olisi kenties voinut liittää osaksi ohjelmaa. Kalenterin käyttöliittymä oli erityisesti huomion kohteena. Yllättäen kuitenkin suurin osa kalentereista oli maksullisia tai muuten opinnäytetyöhön täysin ylimitoitettuja. Ohjelmoija päätti käydä läpi vanhat ohjelmointiprojektinsa, sillä hän oli aikaisemmin luonut kalenterin erääseen projektiin erään opetussivuston pohjalta. Tämän kalenterin pohjalta alettiin luoda kalenteria. Apuna olivat sivusto, jossa oli ohjeet yksinkertaisen kalenterin luomiseksi. Kalenterin runko toteutettiin sivuston ohjeiden mukaan, minkä jälkeen lähdettiin muokkaamaan sitä tarpeeseen sopivaksi.

Sivuston kalenteri ei sisältänyt käyttäjän antamien syötteiden käsittelemistä, varattujen päiväysten näyttämistä eri tavalla käyttäjälle, tietokantaan tallennusta tai muuttujien luomista, joten nämä muutokset oli toteutettava.

6.2 Tietokanta

Tietokannan suunnittelussa oli otettava huomioon, että luotavalla moduulilla pitäisi olla mahdollisuus eteenpäin myyntiin. Tämä tarkoitti sitä, että tietokanta pitäisi rakentaa helposti muokattavaksi, jottei jokaisessa mahdollisessa myyntitilanteessa jouduttaisi suunnittelemaan koko tietokantaa uudestaan.

Ensimmäisessä toteutusvaiheessa pyrittiin tarkentamaan asiakkaan tarpeita ja kartoittamaan, mitä kaikkia liike-elämään ja palveluun liittyviä tietoja tarvittaisiin. Tämän jälkeen alettiin suunnitella tietokantaa piirtämällä siitä alustavia kaavioita ja miettimällä tietueiden tyyppejä sekä pituuksia. Mietittiin myös, minkälaisia relaatioita kanta tulisi käyttämään, ja tarvittaisiinko mahdollisia aputauluja ja kuinka paljon. Palveluun ja tilaamiseen liittyviä tietoja tarkennettiin valokuuvamolta ja päädyttiin siihen tulokseen, ettei tietokannan toteutuksen pohjalta ole tarvetta moni-moneen relaatiolle. Tämä teki tietokannan suunnittelun sekä toteutuksen huomattavasti nopeammaksi ja helpommaksi, kuin mitä alunperin oli ajateltu.

Tietokannan valmistuttua ohjelmoija oli valmis alustavasti testaamaan kantaa hyödyntämällä aikaisemmin luotua kalenterifunktiota. Ainoina pieninä haittoina ilmeni, että tietokannan päiväys sekä aikatietueet vaativat kantaan syötettävät tiedot tietyissä formaateissa. Päiväyksen tallentavissa tietueissa päiväyksen tuli olla amerikkalaisessa muodossa, joten asia korjattiin kirjoittamalla muutama rivi koodia ohjelmaan, jolla pystyttiin pilkkomaan sekä uudelleen muodostamaan päiväys muotoon, joka kelpaisi tietueille. Aikaa käsittelevät tietueet olivat puolestaan muotoa hh:mm:ss, joten käytettiin samaa metodia kuin päiväysten kanssa ja generoitiin automaattisesti varattujen aikojen loppuun sekunnit sekä väleihin kaksoispisteet. Myöhemmin asiakasrekisterin tulostusta luotaessa lisättiin koodiin vielä kohta, joka poistaa sekunnit tietokantaan tallennetusta ajasta, kun se näytetään käyttäjälle.

6.3 Asiakasrekisteri

Ajanvarausrekisterin suunnittelu aloitettiin kartoittamalla tilaajan tarpeita. Keskeiseksi tarpeeksi muodostui nähdä kaikki oleelliset tapahtumatiedot yhdellä vilkaisulla. Lisäksi haluttiin muodostaa muutamia erilaisia näkymiä, joiden tarkoituksena olisi lisätä asiakasrekisterin hallittavuutta. Hallittavuudella tarkoitettiin sitä, että käyttäjä pystyisi helposti listamaan kuluvan kuun tapahtumat, tulevat ja menneet tapahtumat. Näiden tietojen pohjalta alettiin suunnitella ja toteuttaa rekisteriä.

Ensimmäisenä osana oli suunnitella ja toteuttaa sisäänkirjautuminen. Sisäänkirjautuminen toteutettiin yksinkertaisesti luomalla tietokantaan yksinkertainen taulu, joka pitää sisällään käyttäjätunnuksen ja salasanan. Tämän jälkeen siirryttiin luomaan tarvittava lomake, lomakkeen tietojen sanitointi sekä kysely, jolla verrattiin syötteitä kannassa oleviin. Kun salasanaa verrataan kannassa olevaan, käännetään se md5-muotoon, ja siihen lisätään suolaus. Olettaen että käyttäjätunnus ja salasana ovat oikein, luodaan sessiomuuttuja. Sessiomuuttujaa käytetään kyselyitä suoritettaessa varmentamaan, että käyttäjällä on oikeus selata asiakasrekisteriä. Jos käyttäjän salasana tai käyttäjätunnus ei vastaa kannassa olevaa, ilmoitetaan hänen yhdistelmänsä olevan virheellinen.

Tiedot, joita asiakasrekisterissä säilytetään, eivät ole ”tärkeitä”, joten minkäänlaista tunnistusta hyökkäyksien varalle ei ole ohjelmaan tehty. Tämän jälkeen siirryttiin toteuttamaan ja tarkentamaan asiakkaan tarpeita asiakasrekisterin listausten suhteen. Neuvottelujen jälkeen päädyttiin toteuttamaan listaukset, jotka käsittäisivät seuraavat kokonaisuudet: kaikki asiakkaat, tämän kuun tapahtumat, tulevat tapahtumat, menneet tapahtumat sekä mahdollista jatkokehitystä varten maksaneet asiakkaat ja maksamattomat asiakkaat. Kaikki listaukset perustuvat tietokantakyselyihin, joiden parametrit vaihtelevat valitun listauksen mukaan. Viimeisenä toteutettavana asiana oli uloskirjautuminen, joka toimi tuhoamalla käynnisä olevan session ja ohjaamalla asiakkaan takaisin sisäänkirjautumissivulle.

7 OHJELMISTORAKENTEEN KUVAUS SEKÄ KAAVIOT

7.1 Ajanvarausjärjestelmän ja asiakasrekisterin dokumentaatio

Ajanvarausjärjestelmän ja asiakasrekisterin toimintoja on kuvattu prosessikaavioilla (liitteet 2 ja 3) sekä luokkadiagrammeilla (liitteet 4 ja 5). Luokkadiagrammi on kokonsa tähden vielä ”selitetty auki” luvussa 7.2 Tiedostokuvaus. Vaikkei PHP-koodissa ole yleensä tapana tyyppiluokitella muuttujia, on prosessikaaviossa käytetty tyyppiluokittelutapaa, jolla pyritään tarjoamaan dokumentaatioon parempi ymmärrys siitä, millaista tietoa kyseiset muuttujat pitävät sisällään. Tällä pyritään tuomaan luokkadiagrammiin lisää selkeyttä ja kuvata, miten muuttujat liittyvät kunkin kohdan toimintoihin.

7.2 Tiedostokuvaus: Kalenteri

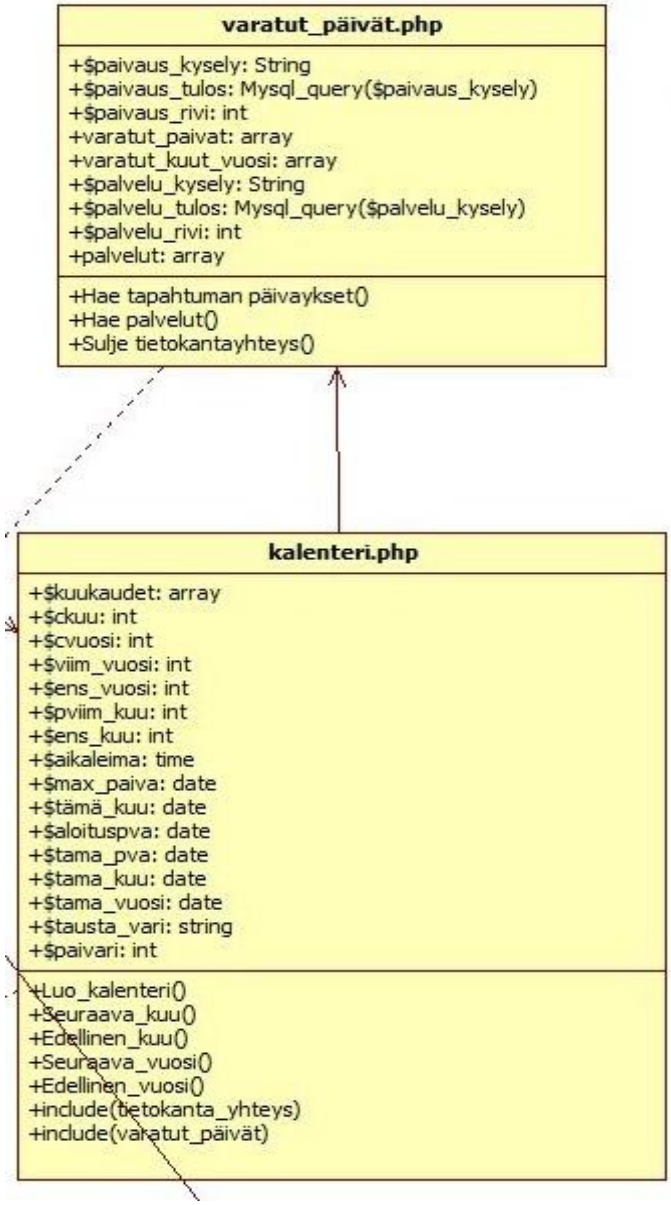
tilaus.php toimii päätiedostona, johon liitetään muita osakokonaisuuksia, moduuleita ja kutsuu funktioita. tilaus.php huolehtii session aloituksesta, liitettävien tiedostojen palvelinpolun määrityksistä, tiedostojen liittämisistä, ehtolauseista, virhetilanteiden sessiomuuttujien luonnista/tuhoamisesta ja session lopettamisesta (kuva 8).

Kuva 8. tilaus.php:n päätoiminnot.

kielivalikko.php sisältää kaikki otsikot, tulosteet ja varoitukset mitä käyttäjä ruudullaan voi nähdä (kuva 9). Kielivalikko mahdollistaa nopean tavan muuttaa tulosteita ja muuttaa järjestelmä monikieliseksi ilman lähdekoodin muokkaamista.

Kuva 9. kielivalikko.php: Vastaa teksteistä ja otsikoista, joita käytetään käyttäjille tulostettavissa lomakkeissa.

kalenteri.php vastaa kalenterin luonnista ja kaikista kalenterin tarvitsemien muuttujien tai tietojen alustuksista sekä käsittelyistä (kuva 10). *varatut_päivät.php* suorittaa varattujen päiväysten kyselyn, jotta *kalenteri.php* pystyy käsittelemään ne tietokantaan avatun tietokantayhteyden *tietokanta_yhteys*-tiedoston kautta.



Kuva 10. kalenteri.php ja varatut_päivät.php muuttujineen ja toimintoineen.

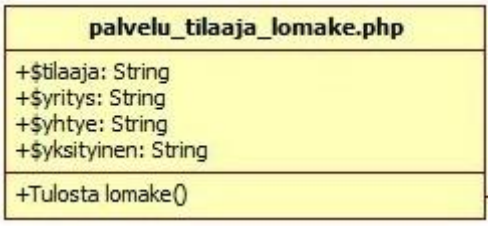
päiväys_käsittelijä.php muodostaa käyttäjän valitsemasta päiväyksestä tietokantahakuihin kelpaavan muuttujan (kuva 11).

Kuva 11. päiväys_käsittelijä.php muuttujineen ja toimintoineen.

tietokanta_varaus_tarkistus.php ottaa yhteyden tietokanta_yhteys-tiedoston avulla tietokantaan ja suorittaa kyselyn, jolla tarkistetaan, onko käyttäjän valitsema päiväys vapaa (kuva 12).

Kuva 12. tietokanta_varaus_tarkistus.php muuttujineen ja toimintoineen.

palvelu_tilaaja_lomake.php tulostaa tarvittavan lomakkeen, johon käyttäjä antaa tiedon edustamastaan tahosta (kuva 13).

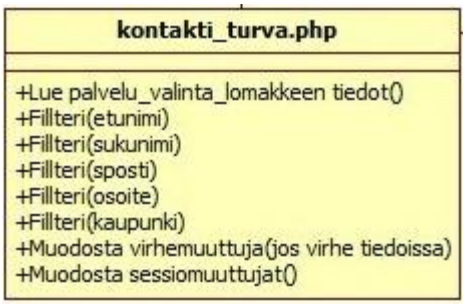


Kuva 13. palvelu_tilaaja_lomake.php muuttujineen ja toimintoineen.

kontakti_lomake.php näyttää käyttäjälle lomakkeen, johon täytetään kontaktihenkilön tiedot, sekä tieto siitä, millaiseen tilaisuuteen kuvaaja halutaan varata (kuva 14). Tämän lisäksi lomake luo sessiomuuttujat annetuista tiedoista, jotka sanitoidaan ja kirjoitetaan tietokantaan myöhemmin.

Kuva 14. kontakti_lomake.php muuttujineen ja toimintoineen.

kontakti_turva.php jos kontakti_ lomake on täytetty kokonaan siirrytään kontakti_turvaan, joka sanitoi käyttäjän syötteet, käyttämällä filtteri funktioita ja luo sanitoiduista tiedoista sessiomuuttujat. Mikäli tiedot ovat olleet vääränlaisi, luodaan virhemuuttuja, jota käytetään ilmoittaan käyttäjälle virheellisistä tiedoista. Tämän lisäksi käyttäjä joutuu täyttämään lomakkeen uudestaan (kuva 15).



Kuva 15. kontakti_turva.php toimintoineen.

palvelu_valinta_lomake.php tulostaa käyttäjälle lomakkeen, johon käyttäjä syöttää osoitteen missä tapahtuma järjestetään, kaupungin, tapahtuman aloitusajan ja arvioidun päättymisajan (ei pakollinen). Tämän lisäksi lomake tulostaa aikaisemman tapahtuma valinnan perusteella (kontakti_lomake) lomaakkeeseen painikkeet, joilla voi valita haluamansa kuvauspalvelut tapahtumaan. Mikäli käyttäjä on valinnut palvelukseen muu kohdan kontakti_lomakkeessa tulostetaan käyttäjälle painikkeiden sijaan tekstilaatikko, johon käyttäjä voi antaa lyhyen kuvauksen tarvitsemistaan palveluista ja tapahtuman yleisestä tyylistä (kuva 16).

Kuva 16. palvelu_valinta_lomake.php muuttujineen ja toimintoineen.

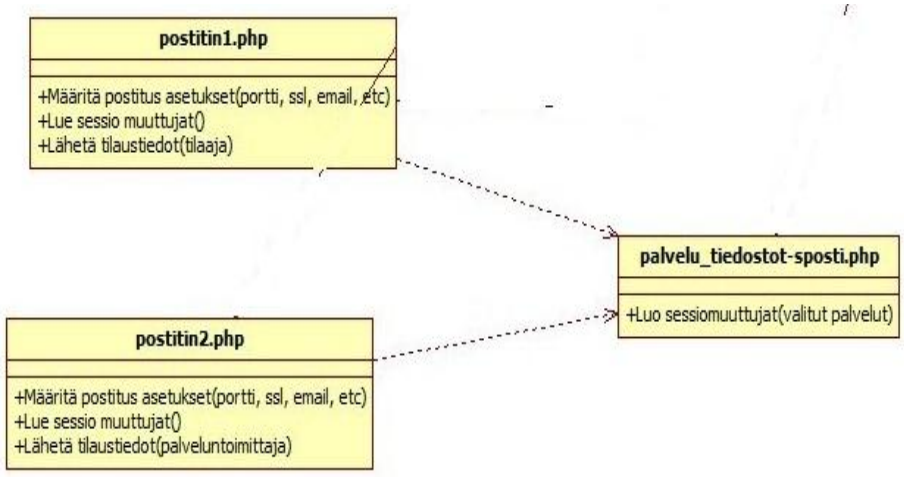
palvelu_turva jos *palvelu_valinta_lomake* on kokonaan täytetty, *palvelu_turva lukee* annetut syötteet, sanitoi ne filtterifunktioiden avulla sekä luo niistä sessiomuuttujat (kuva 17). Mikäli annettu tieto on ollut virheellistä luodaan virhe muuttuja, jonka avulla käyttäjälle ilmoitetaan virheellisestä tiedosta sekä pyydetään häntä täyttämään palvelu_valinta_lomake uudestaan.

Kuva 17. palvelu_.php toimintoineen.

Mysql_kirjoitus.php suoritetaan jos kontakti_lomake, palvelu_valinta_lomake on täytetty ja palvelu_turva:ssa, sekä kontakti_turvassa on luotu sessiomuuttujat tiedoille, luetaan sessiomuuttujat, valitaan palvelun mukaiset MySQL kirjoitussyntaksit, kirjoitetaan ne tietokantaan ja suljetaan tietokantayhteys (kuva 18). Mikäli tietokantaan kirjoittaminen epäonnistuu, tulostetaan käyttäjälle virheilmoitus ja pyydetään häntä yrittämään uudestaan.

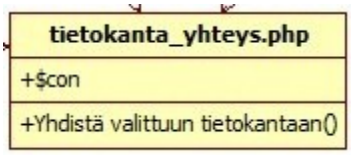
Kuva 18. Mysql_kirjoitus.php toimintoineen.

postitin 1 ja 2 .php lähettävät asiakkaalle ja palvelun tarjoajalle tiedot, joita yritys tarvitsee tietääkseen asiakkaan tarpeet sekä asiakkaalle viestin jossa luetellaan asiakkaan antamat tiedot. *palvelu_tiedostot-sposti.php* luo sessiomuuttujat valituista palveluista, joita postittimet käyttävät (kuva 19).



Kuva 19. *postitin* 1, 2 .php ja palvelu_tiedostot-sposti.phpn relaatiot ja toiminnot.

tietokanta_yhteys.php muodostaa valittuun tietokantaan yhteyden antamalla käyttäjätunnukset, salasana ja valitun tietokannan nimen $con muuttujassa (kuva 20).



Kuva 20. tietokanta_yhteys.php muuttujineen ja toimintoineen.

7.3 Tiedostokuvaus: Ajanvarausjärjestelmä

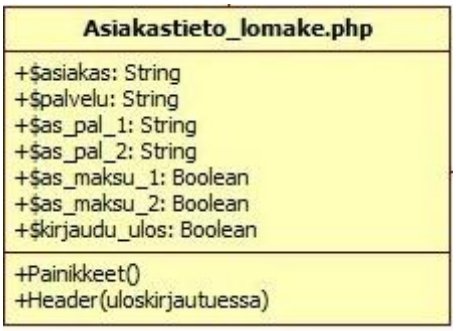
Sisaankirjaudu-tiedosto määrittää tarvittavat palvelinpolut, liittää tarvitsemansa tiedostot, suorittaa ehtolauseet, aloittaa session, kun sisäänkirjautuminen on onnistunut, luo sisäänkirjautumisesta sessiomuuttujan ja tulostaa sisäänkirjautumislomakkeen (kuva 21).

Kuva 21. Sisaankirjaudu.php muuttujineen ja toimintoineen.

Tietokanta_tarkistus-tiedosto käsittelee käyttäjätunnuksen ja salasanan muokkaamalla ne tietoturvallisiksi. Käyttäjätunnus muutetaan tietokantaturvalliseksi ja salasana käännetään salattuun muotoon, jolloin sitä voidaan verrata tietokannassa sijaitsevaan. Tietokanta_tarkistus suorittaa salasana ja käyttäjätunnus–pari kyselyn avaamalla yhteyden tietokantaan, suorittamalla kyselyn ja sulkemalla tietokantayhteyden (kuva 22)

Kuva 22. Tietokanta_tarkistus.php muuttujineen ja toimintoineen.

Asiakastieto_lomake.php tulostaa tarvittavat tietojen listaukseen liittyvät painikkeet, sekä huolehtii uloskirjautumisen yhteydessä asiakkaan ohjaamisesta pääsivulle (kuva 23).

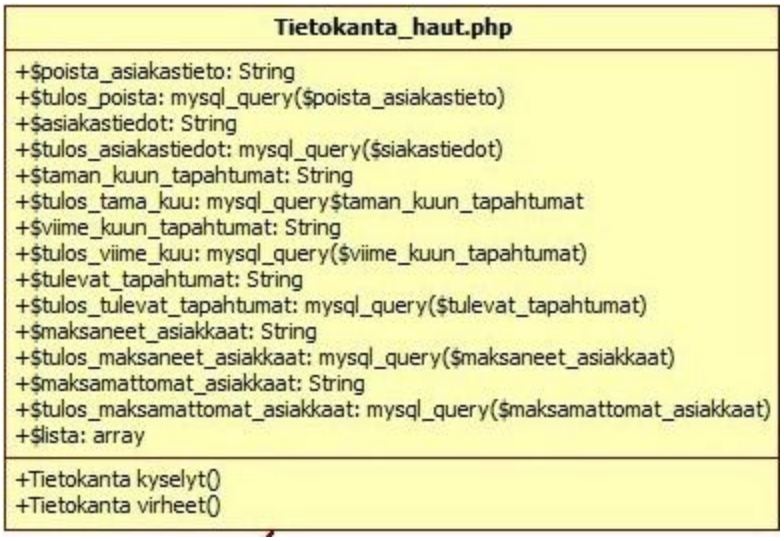


Kuva 23. Asiakastieto_lomake.php muuttujineen ja toimintoineen.

Listari.php huolehtiii asiakastietojen lomakenäkymästä. Listari asettelee tarpeelliset tiedot valitun tietojen listaamistavan mukaisesti tulostusnäkymään (kuva 24).

Kuva 24. Listari muuttujineen ja toimintoineen.

Tietokanta_haut.php huolehtii tietokannassa suoritettavista hauista asiakkaan listausnäkymän mukaisesti ja hoitaa tietokantavirheiden käsittelyn (kuva 25).



Kuva 25. Tietokanta_haut.php muuttujineen ja toimintoineen.

7.4 Tietokantakuvaus

Tietokantaa lähdettiin suunnittelemaan sillä perusteella, että sitä on mahdollista muuttaa helposti sekä muokata erilaisiin tarpeisiin mahdollista ohjelman kaupallistamista varten. Tietokantamoottoriksi valittiin InnoDB, koska sillä on monia hyviä puolia MyISAMiin verrattaessa. Päätös tehtiin seuraavien tietojen pohjalta:

- InnoDB on uudempi
- InnoDB lukitsee rivit kirjoitettaessa tietokantaan taulujen sijaan
- Viite-eheysavaimia ei ole käytössä MyISAM:issa
- InnoDB on helpompi ja varmempi kuin MyISAM palautustilanteissa, joissa järjestelmä on kaatunut.

Palvelun toteutuksessa päätettiin, että asiakkaan tavoittamiseksi riittäisi yksi puhelinnumero ja sähköpostiosoite eikä näin ollen tietokannan normalisoinnin (4NF) kaikkia ehtoja täytetty. Neljännnen normaalimuodon ehtojen täyttämättä

jättäminen perusteltiin lisäksi tietokannan käytettävyyden sekä mahdollisten jälleenmyyntiin tarvittavien muutosten helpottamiseksi.

Ensimmäinen normaalimuoto täytetään, koska kaikki taulujen sarakkeiden attribuuttien arvot ovat atomisia. Tämä toteutuu, vaikkakin yleisesti kontaktitiedot eivät ole atomisia. Tässä tapauksessa kuitenkin asiakkaaseen liitetään vain yksi puhelinnumero ja sähköposti, koska asiakkaan tavoittaminen jatkuvasti ei ole tärkeää.

Toinen normaalimuoto toteutuu, koska taulujen avaimia ei ole valittu avainehdokkaista, vaan ne toteutetaan keinotekoisesti autoinkrementoidulla avaimella. Tällöin kaikki muut taulun tiedot ovat riippuvaisia tästä identifikaationumerosta.

Kolmas normaalimuoto täytetään, koska tietueiden kentät eivät ole riippuvaisia muista kuin avaimista. Toisin sanoen kun tietoa poistetaan, ei menetetä mitään muuta kuin avainattribuutista riippuvia tietoja.

Tietokanta ei vaatinut yksi-moneen tai moni-moneen relaatiota, koska asiakas ei ollut kiinnostunut pakottamaan käyttäjiä luomaan profiilia sivuilleen. Tästä seurasi se, että jokaisen tilauksen yhteydessä asiakas antaa tietonsa uudelleen, vaikka ne löytyisivätkin jo tietokannasta. Tarkoituksena oli luoda moduuli, jolla voitaisiin helposti vastata tilauksiin automaattisesti, selata asiakastietoja ja laskuttaa niiden perusteella.

Tietokanta koostuu seuraavista tauluista (skandinaaviset merkit on tarkoituksella jätetty tietokannan kenttien nimistä pois). Kentät esitellään muodossa Tietueen_nimi(TYYPPI). (Liite 1).

- asiakas_tiedot
- yritys_tiedot
- yhtye_tiedot
- tapahtuma_tiedot
- kastetilaisuus
- haat
- rippijuhlat

- syntymapaivat
- haat
- muu
- sisäänkirjautuminen

7.5 Postitustoimintojen kuvaus

Sähköpostitoimintoja varten on PHP:ssä oma postitinfunktionsa, mutta sitä on mahdotonta ajaa ainoastaan lokaalisti toimivassa järjestelmässä. Ennen koodin siirtoa palvelimelle päätettiin varmistaa, ettei postitinfunktiosta tulisi ongelmaa enää palvelimelle siirtovaiheessa. Vaihtoehtoiseksi ratkaisuksi etsittiin ulkopuolinen postitinfunktio, jolla pystyi testaamaan postituksen ja käyttämään sitä, jos ongelmatilanteita olisi ilmennyt. PHP:n oma postitusfunktio on osa PHP:n asennusta eikä tarvitse erillisiä lisäosia toimiakseen. PHP:n oman postitusfunktion voi ottaa käytöön konfiguroimalla **php.ini-**tiedostoa. Tiedostossa Windows-laitteissa pitää antaa SMTP-palvelimen DNS-nimi tai ip-osoite, smtp_port (portti, jota käytetään liikennöintiin) sekä sendmail_from, johon määritellään osoite, jota käytetään sähköpostien lähettämiseen. Unix-laitteissa pitää määritellä polku (sendmail_path), josta postitusohjelma (sendmail) löytyy. Postitusfunktion syntaksi on muotoa: posti (kenelle, asiaotsikko, viesti, lisäotsikko, parametrit). Näihin syötetään vastaanottaja(t) (pakollinen), mitä viesti koskee (pakollinen), viesti (pakollinen), lisäotsikko (valinnainen) ja mahdolliset lisäparametrit (valinnainen).

Ulkopuolinen postitusohjelma oli nimeltään phpmailer, joka toimii syntaksiltaan PHP mail funktion tavoin (kuva 26), mutta ohjelmassa pystyy määrittelemään erillisen sähköpostin (esim. gmail), johon se muodostaa yhteyden ja käyttää kyseistä sähköpostia viestien lähtettämiseen. Tätä varten pitää määritellä sähköpostin käyttäjätunnus sekä salasana, jolla ohjelma pystyy ottamaan kiinni käytetyn sähköpostisivuston palvelimeen ja käyttämään tätä sähköpostien lähettämiseen.

```
//php.ini extension=php_openssl.dll avattu
    require($_SERVER['DOCUMENT_ROOT'].'/phpmailer/class.phpmailer.php');
    require($_SERVER['DOCUMENT_ROOT'].'./phpmailer/class.smtp.php');
    $mail  = new PHPMailer();

    $mail->IsSMTP();
    $mail->SMTPDebug = 2;                   // enables SMTP debug information (for testing)
    $mail->SMTPAuth  = true;                #enable SMTP authentication
    $mail->SMTPSecure = "ssl";              #sets the prefix to the server
    $mail->Host  = "smtp.gmail.com";        #sets GMAIL as the SMTP server
    $mail->Port       = 465;                #set the SMTP port
    $mail->Username   = [redacted]                          #your gmail username
    $mail->Password   = [redacted]          #Your gmail password
    $mail->From       = "houhouhoukku@gmail.com";           #your gmail id
    $mail->FromName   = "Visual Vortex";                    #your name
    $mail->Subject    = "Ajanvaraus: ".$_SESSION['fname']. " ".$_SESSION['lname'];
    $mail->WordWrap   = 50;
    $mail->AddAddress("houhouhoukku@gmail.com");
    $mail->IsHTML(true); // send as HTML
    $mail->Body = "<b>Ajanvaraus visual vortex </b> <br>

<br> <b>Kontaktihenkilön tiedot:</b> <br> <b>Nimi:</b> ".$_SESSION['fname']." ".$_SESSION['lname']." <br>
<b>Sähköposti:</b> ".$_SESSION['email']."<br>
<b>Puhelinnumero:</b> ".$_SESSION['pnro']."<br>
<b>Osoite:</b> ".$_SESSION['address']." ".$_SESSION['tapahtuma_kaupunki']."<br><br>
```

Kuva 26. Esimerkki PHP mail funktion parametreista ja käytöstä.

8 TIETOTURVA

Tietoturvan osalta noudatetaan seuraavia periaatteita (luku 3):

- Luottamuksellisuus
- Eheys
- Pääsynvalvonta.

Luottamuksellisuus toteutettiin samalla tavalla kuin pääsynvalvontakin. Ainoastaan palvelimelle pääsyn omaava henkilö, joka tietää tietokannan salasanan, voi luoda uusia käyttäjiä, jotka näkevät tilaajien tiedot. Tämä toteutustapa on luotu, koska yritys ei tarvinnut kuin yhden käyttäjän tarkistamaan asiakkaiden tietoja. Tämän takia turvallisimmaksi vaihtoehdoksi katsottiin, että sivuston ylläpitäjä tai muu luotettava tarpeellisen teknisen tiedon omaava taho olisi vastuussa mahdollisista käyttäjätunnus-salasanaparien lisäämisistä. Tällä tavoin voitiin täysin ehkäistä mahdollisuuden ”tyhjien” tai liian heikkojen käyttätunnusten tai salasanojen luomiseen. Luottamuksellisuuden tarve ei ole kuitenkaan kovin korkea tällaisessa palvelussa. Suurin käyttäjiin liittyvä tarve on, ettei heidän tilauksensa tapahtumapaikka ja aika saisi tulla esille. Tämä on otettu huomioon eriyttämällä asiakasrekisteri omaksi kokonaisuudekseen, joka on vain valitun henkilön nähtävillä. Lisäksi on pyritty estämään mahdolliset koodi-injektiot filtteröimällä ja poistamalla ohjelmakoodiin viittaavat sanat asiakkaiden syötteistä ennen kuin suoritetaan mitään tietokantaoperaatioita.

Eheyttä pyrittiin valvomaan normalisointi muotojen avulla sekä myöhemmin tarkemmin kuvattujen testausten kautta. Näin varmistettiin, että tiedot, jotka tallennetaan tietokantaan pysyvät yhdenmukaisena huolimatta lisäyksistä, muokkauksista tai poistamisesta.

Vaikka XAMPP on oletusarvoisesti kaikille auki ja suuri osa turvallisuusominaisuuksistakin on pois päältä, näin se ei ole palvelimella toteutetussa versiossa. Palvelinversion PHP.ini tiedosto on oletuksena konfiguroitu turvallisuden takaavaksi, ja sitä eivät pääse muokkaamaan kuin

palvelimen ylläpito. Pääsynvalvonta on toteutettu session muuttujille, joihin tallennetaan käyttäjän oikeus selata ja poistaa tietoja, mikäli hänen salasanansa ja käyttäjätunnuksensa on todennettu tietokannasta. Muussa tapauksessa ohjelma ohjaa käyttäjän kirjautumissivulle.

Osapuolten todentamista sekä kiistämättömyyden mahdollisuutta ei moduulissa toteuteta. Tämä perusteltiin järjestelmän pienuudella sekä sillä, ettei järjestelmällä voi aiheuttaa haittaa toisille henkilöille, koska tilaukset eivät toteudu ennen kuin tilaaja on maksanut varausmaksun yrityksen tilille.

8.1 Käyttäjäsyötteiden sanitointi

Käyttäjäsyötteet sanitoidaan mahdollisen ohjelmalle tai tietokannalle vahingollisen koodin löytämiseksi ja vaarattomaksi muuttamisen takia. PHP 5.4.0:sta eteenpäin ”**magic quotes**” käsky on poistettu, ja muut koodia sanitoivat käskyt on luotu omiksi erillisiksi filttereiksi. Lisäksi kaikki tiedot muunnetaan ”mysql_real_escape” käskyllä tietokantaturvallisiksi vielä filtteröinnin jälkeen.

Näitä filttereitä käyttämällä pystytään säästämään aikaa koodauksessa ja helpottamaan ohjelman turvallisuuden varmistusta. Aikaisemmin internetissä oli paljon erilaisia tapoja sanitoida PHP:ssä käyttäjäsyötteitä, ja oli vaikeaa tietää, mitkä näistä tavoista olivat luotettavia sekä mahdollisimman aukottomia. Nykyään filttereillä on aikaansaatu eräänlainen standardi, jolla on helppo taata hyvä tietosuoja nopeasti. Tosin tietyissä tilanteissa on tarpeellista käyttää vanhoja menetelmiä, kuten koodisanojen poistamista sekä tietokantaan liittyvien sanojen muokkaamista. Edellisestä esimerkkinä voidaan käyttää alfanumeerisia yhdistelmiä.

Itse ohjelmassa on käytetty seuraavia filttereitä:

- FILTER_SANITIZE_STRING
- FILTER_SANITIZE_NUMBER_INT
- FILTER_SANITIZE_EMAIL

- FILTER_VALIDATE_EMAIL.

String-suodatin varmistaa, että käyttäjän syöte sisältää vain kirjaimia eikä syöte sisällä ohjelmakäskyjä. Lisäksi on mahdollista määritellä suodatin poistamaan tai enkoodaamaan tiettyjä ASCII taulun merkkejä. INT-suodatin toimii samoin kuin string-suodatinkin, mutta poistaa kaikki merkit, jotka eivät täytä seuraavia ehtoja: numeerinen, + tai -. SANITIZE_EMAIL poistaa käyttäjän syötteestä kaikki, paitsi alfanumeeriset merkit sekä *!#$%&'*+-/=?^_`{|}~@.[]* merkit. VALIDATE_EMAIL puolestaan varmistaa, että annettu sähköposti on oikean muotoinen, eikä vain jonkinlainen käyttäjän antama sarja merkkejä.

```
2
3       //filtterit ovar php:omia funktioita string,email ja numeerisien muuttujien sanitoimiseksi
4       if (isset($_POST['Submit'])) {
5
6
7           if ($_POST['fname'] != "") {
8               $firstname = filter_var($_POST['fname'], FILTER_SANITIZE_STRING);
9
10          } else {
11
12              $_SESSION['error'] = 1;
13          }
14          if ($_POST['lname'] != "") {
15              $lastname = filter_var($_POST['lname'], FILTER_SANITIZE_STRING);
16              }
17           else {
18          $_SESSION['error'] = 1;
19          }
20              if (isset($_POST['email'])) {
21          $email = filter_var($_POST['email'], FILTER_SANITIZE_EMAIL);
22          if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
23              $_SESSION['error'] = 1;
24          }
```

Kuvio 27. Esimerkki filtterien käytöstä.

8.2 Tietokantaturvallisuus

Tietokannan turvallisuus palvelimella on toteutettu luomalla tietokannalle oma käyttäjätunnus sekä salasana. Ne tarvitaan palvelimen käyttäjätunnuksen ja salasanan lisäksi, jos haluaa tehdä muutoksia, tai selata kantaa muuten kuin

MySQL-injektioiden avulla, jotka on pyritty estämään edellisessä kappaleessa esitetyin käyttäjäsyötteiden suodattamisella.

Tietokannassa itsessään ei ole kovinkaan ”arvokasta” tietoa, joten erilliseen salaamiseen ei ollut tarvetta. Ainoastaan tietokannassa olevat asiakasrekisterin käyttäjätunnus ja salasana ovat sellaisia. Näiden salaamiseen on käytetty MD5 salausta, joka itsessään ei ole kovinkaan vaikea murtaa, mutta kun salaukseen on lisätty suolaus se muuttaa jo salauksen riittäväksi ohjelmaan. Suolauksella tarkoitetaan yksilöllisen alfanumeerisen yhdistelmän lisäämistä jokaisen luodun salasanan eteen ja loppuun. Tämä toimintatapa estää sateenkaaritaulujen käytön. Sateenkaaritauluksi nimitetään nopeaa tapaa/ohjelmaa, joka kääntää kryptatun hash-funktion takaisin selkokieliseksi. Luonnollisestikaan tässä ei voida paljastaa, millä logiikalla ”suolaus” luodaan, tai sen pituuksia.

9 OHJELMISTOTESTAUS

Testaus tehtiin vesiputousmallilla toteutuksen edetessä. Tämä tarkoittaa, että jokainen osa-alue testattiin sen valmistuessa ja uudelleen, kun siihen liitettiin uusi moduuli tai funktio. Näin pystyttiin valvomaan, ettei ohjelmaan jää mitään suuria ongelmia kun sitä siirretään palvelimelle. Tällä tavalla varmistettiin, että vikatilanteissa ei tulisi ongelmaa etsiä virheen aiheuttajaa moduulien ja funktioiden joukosta. Palvelimella ohjelmaa testattiin vielä ohjelman tilaajan sekä ohjelmoijan toimesta. Näin varmistettiin, ettei ohjelmistossa piilisi mitään kriittisiä virheitä, jotka voisivat aiheuttaa ongelmatilanteita joko ohjelman käyttäjille tai ohjelman omistavalle taholle.

Testaamista ei voitu täysin oikeaoppisesti suorittaa aikaisemmin kuvatuin mustalaatikko- ja valkolaatikkometodein, koska ohjelmoija ja palvelun tilaaja olivat ainoat testaukseen pystyvät henkilöt. Testauksessa kuitenkin meneteltiin valkolaatikkotestauksen mukaisin tavoin ja ohjelmasta käytiin läpi kaikki sen mahdolliset silmukat ja tulosteet. Palvelun tilaaja pyrki toteuttamaan mustalaatikkotestauksen, jossa tilaaja antoi ohjelmaan syötteitä ja kokeili ohjelman toimintoja pyrkien varmistamaan, että ne toimivat halutulla tavalla ja syötteiden ulostulot olivat odotetunlaisia. Käytössä ei ollut testaamista helpottavia ohjelmistoja, joilla olisi voinut tehdä esimerkiksi kuormitustestaamista.

9.1 Testaus

Testausdokumentaatio implementoitiin lyhyempänä versiona kuin useimmissa internetistä löytyvissä esimerkeissä tai suosituksissa. Syynä tähän oli se, että opinnäytetyö venyisi liian pitkäksi, jos kaikki testit käsiteltäisiin hyvän testausdokumentaatiotavan mukaan, eikä toimeksiantaja myöskään vaatinut

mitään perusteellista selvitystä. Hän halusi vain nähdä, mitä on testattu, miten on testattu ja mitkä olivat testin tulokset.

9.2 Ajanvarauksen testitapaukset

9.2.1 Testitapaus 1: kalenterin luotettavuus

Tarkoituksena oli testata kalenterin luotettavuus. Eli kun kalenteria selaa eteen tai taaksepäin (kuukausia tai vuosia), tarkistettiin, tulostuvatko kuukausien pituudet oikein, sekä tulostuvatko oikeat päivämäärät oikeille viikonpäiville, ja että testiä suoritettaessa kyseisen päivän päiväys sekä varatut päiväykset on merkitty eri värillä kuin muut päiväykset.

Toteutus

Testi toteutetaan käyttämällä suosituimpia selaimia ja niiden suosituimpia versioita. Jokaisella eri selaimella pyritään samaan lopputulokseen, kuitenkin siten että kriittisiä selaimia ovat firefox ja chrome, koska nämä muodostavat 78,6 % osuuden selainten käytöstä tammikuussa 2013 [14] .

Testissä käytetään ohjelman edellinen ja seuraava painikkeita kuukausissa ja vuosissa siirtymiseen, sekä url:ilta tapahtuvaa suoraa manipulointia, jossa vaihdetaan kuukausien ja vuosien numeroita suoraan. Tämän lisäksi tietokannasta tarkistetaan, mille päiville on tehty varauksia, ja että varatut päivät on merkitty kalenterissa eri värillä.

Testin vaatimukset

Jotta testin tulos on läpäisty, pitää kuukausien pituuksien olla oikeat sekä päiväysten sijaita oikeilla viikonpäivillä vähintään firefox sekä chrome selaimissa ja testin suorittamispäivän sekä varattujen päivien olla eri värillä merkitty. Kaikissa muissa tapauksissa testin tulos on hylätty.

Testinlopputulos

Testi toteutettiin käyttäen suosituimpia selaimia, suluissa versio numero; Firefox 19, Chrome 24,IE 8, Safari 6

Testissä päiväykset tulostuivat oikein kaikilla selaimilla, ja päiväyksien värimerkinnät toimivat. Lisäksi menneiden päivien tunnistus toimi virheviesteineen odotetulla tavalla eikä kalenterista päässyt siirtymään varausvaiheeseen menneellä päiväyksellä millään testin toteutuksen määrittämillä tavoilla. Testi oli siis kaikilta osa-alueiltaan hyväksytty.

9.2.2 Testitapaus 2: kalenterin linkit

Tarkoituksena on testata, että painettaessa kalenterin päiväyksen linkkiä käyttäjä joko ohjautuu seuraavalle sivulle (palveluntilaajalomakkeeseen) tai hänelle ilmoitetaan, että hänen valitsemansa päiväys on jo mennyt tai on varattu jo aikaisemmin.

Toteutus

Kalenterista valitaan sattumanvaraisia vuosia, kuukausia ja päiviä joiden linkkiä painetaan. Ohjelman pitää käyttäytyä ennustetulla tavalla, jolloin ohjelma ohjaa käyttäjän palveluntilaajalomakkeeseen, mikäli valittu päiväys on sekä vapaa että tulevaisuudessa. Sellaisessa tapauksessa, jossa päiväys on jo mennyt tai varattu, pitää käyttäjälle tulla ilmoitus tästä.

Testin vaatimukset

Onnistuakseen pitää käyttäjän saada lomakesivu eteensä vain, jos hänen valitsemansa päiväys on tulevaisuudessa ja avoin. Muussa tapauksessa testin tulos on hylätty.

Testinlopputulos

Testi suoritettiin 15 kertaa/selain. Päiväyksiä manipuloitiin painikkeilla (10 kertaa) ja muokkaamalla URLia. URL-osoitteen muokkaus suoritettiin yrittämällä

manipuloida suoraan vuotta ja/tai kuukautta,sekä kopiomalla suoraan lomakesivun url, ja vaihtamalla sen tiedot menneisiin päiväyksiin. Menneiden päivien tunnistus toimi virheviesteineen odotetulla tavalla ja kalenterista ei päässyt siirtymään varausvaiheeseen menneellä päiväyksellä millään testin toteutuksen määrittämillä tavoilla. Testi oli siis kaikilta osa-alueiltaan hyväksytty.

9.2.3 Testitapaus 3: palveluntilaaja-lomake

Kun käyttäjä valitsee palveluntilaaja-lomakkeessa yritys, yhtye tai yksityishenkilövaihtoehdon, käyttäjä ohjaantuu kontaktihenkilölomakkeeseen.

Toteutus

Valitaan jokaisella selaimella 10 kertaa jokainen vaihtoehto ja tarkistetaan, ettei mitään odottamatonta ilmene.

Testin vaatimukset

Jos tapahtuu jotain muuta, kuin siirtyminen kontaktilomakkeeseen, on testi hylätty. Ennen testin suorittamista täytyy olla Testitapaus 1 suoritettu.

Testinlopputulos

Testi toteutettiin 20 kertaa jokaiselle vaihtoehdolle niiin mustalaatikko, kuin valkolaatikko testauksenkin tapoja noudattaen, eikä odottamattomia tilanteita ilmennyt. Testi hyväksytty

9.2.4 Testitapaus 4: kontaktihenkilö-lomake

Kontaktihenkilö-lomakkeen tietojen täyttäminen ja sanitointi tietokantaa varten. Testataan mysql-injektioita sekä php & html-koodin syöttämistä kenttiin ja varmistetaan, etteivät ne voi aiheuttaa haittaa tietokannalle tai ohjelmalle. Lisäksi varmistetaan, että erikoismerkit sekä numerot/kirjaimet sanitoidaan niistä kentistä, joihin ne eivät kuulu, ja annetaan virheilmoitus mikäli käyttäjä syöttää kelpaamattomia merkkejä kenttiin.

Toteutus

Yritetään syöttää haitallisia php-, html- ja mysql-koodeja järjestelmään lokaalissa ympäristössä, jolloin järjestelmä on helppo palauttaa mahdollisista haittatilanteista. Sen lisäksi syötetään erikoismerkkejä ja tarkistetaan virheilmoitukset sekä tietokanta ja luetaan, mitä syötteitä tietokantaan päätyy. Ennen testin suorittamista täytyy olla Testitapaus 1 & 2 suoritettuina.

Testin vaatimukset

Ohjelma ei saa päästää käyttäjää haitallisen koodin avulla aiheuttamaan ongelmatilanteita tietokannassa, näkemään tai vaikuttamaan asiakastietoihin , eikä aiheuttamaan virhetilannetta, jossa järjestelmä joudutaan palauttamaan käyttäen varmuuskopioita.

Testin lopputulos

Testi toteutettiin, eikä havaittu haitallisten koodien pystyvän aiheuttamaan järjestelmälle vahinkoa. Testin lopputulos täytyy kuitenkin ottaa vastaan pienellä varauksella, siitä syystä ettei testissä ollut mahdollista saada varsinaista asiantuntija-apua. Testi hyväksytty.

9.2.5 Testitapaus 5: tapahtumatiedot-lomake

Tapahtumatiedot-lomakkeen tietojen täyttäminen ja sanitointi tietokantaa varten. Testataan mysql injektioita sekä php & html –koodin syöttämistä kenttiin ja varmistetaan, etteivät ne voi aiheuttaa haittaa tietokannalle tai ohjelmalle. Lisäksi varmistetaan, että erikoismerkit sekä numerot tai kirjaimet sanitoidaan niistä kentistä, joihin ne eivät kuulu, ja annetaan virheilmoitus mikäli käyttäjä syöttää kelpaamattomia merkkejä kenttiin.

Toteutus

Yritetään syöttää haitallisia php, html ja mysql –koodeja järjestelmään lokaalissa ympäristössä, jolloin mahdolliset haittatilanteista on helppo palauttaa järjestelmä. Sen lisäksi syötetään erikoismerkkejä ja tarkistetaan virheilmoitukset sekä tietokanta ja luetaan, mitä syötteitä tietokantaan päätyy. Testitapaus 1, 2, 3 pitää olla suoritettuina ennen.

Testin vaatimukset

Ohjelma ei saa päästää käyttäjää haitallisen koodin avulla aiheuttamaan ongelmatilanteita tietokannassa, näkemään tai vaikuttamaan asiakastietoihin, eikä aiheuttamaan virhetilannetta, jossa järjestelmä joudutaan palauttamaan käyttäen varmuuskopioita. Hetkellinen ohjelman kaatuminen tai virhetilanteet (haitallisen koodin syöttäjän selaimessa), jotka eivät hankaloita muiden käyttäjien toimintaa, voidaan vielä sallia.

Testin lopputulos

Virheilmoitukset sekä erikoismerkkien sanitointi toimi odotetusti, eikä minkäänlaista ongelmatilannetta ilmennyt. Testin lopputulos täytyy kuitenkin ottaa vastaan pienellä varauksella, siitä syystä ettei testissä ollut mahdollista saada varsinaista asiantuntija-apua. Testi hyväksytty.

9.2.6 Testitapaus 6: postitusfunktioiden toiminnallisuus

Suoritetaan sekä SMTP-palvelin funktiolle että phpmailer-funktiolle

Postitusfunktioiden varmennus. Tarkistetaan, että postitusfunktiot käynnistyvät, poimivat tarvittavat tiedot sekä lähettävät ne eteenpäin. Testitapaus neljässä toteutetun dokumentin avulla verrataan että tiedot, jotka ovat lähteneet asiakkaalle sekä toimittajalle, ovat oikein ja kattavat kaiken tarvittavan informaation.

Toteutus

Luodaan 6 erillistä ilmaissähköpostia, joihin palvelu lähettää tiedot lomakkeen täyttämisen lopuksi. Palveluntarjoajan puolesta käytetään vain yhtä lähettävää sähköpostia. Testitapaus 4 pitää olla suoritettuna ennen.

Testin vaatimukset

Kummassakin tapauksessa (SMTP & phpmailer) pitää sähköpostiviestien rakenteen olla selkeä, helposti luettava ja niiden tulee sisältää kaikki niin asiakkaalle kuin toimittajallekin palveluntilaajan tärkeäksi määrittelemät tiedot.

Testin lopputulos

Phpmailer-funktioille testi pystyttiin ajamaan vain lokaalista järjestelmästä, siitä syystä että palvelimen omistavan tahon kieltäytymisestä avaamaan tarvittavia ominaisuuksia php.ini tiedostosta. Lokaalissa versiossa testi oli täysin onnistunut. Phpmailer-funktio hylättiin aikaisemmin mainitun syyn takia ja sitä ei oteta mahdolliseksi rinnakkaisjärjestelmäksi.

SMTP-postitus toimi palvelimella vaatimusten mukaan, eikä ongelmia ilmennyt.

9.3 Asiakasrekisterin testitapaukset

9.3.1 Testitapaus 1: login-lomakkeen tietojen sanitointi

Testataan login-lomakkeen tietojen täyttäminen ja sanitointi tietokantaa varten. Testataan mysql injektioita sekä php & html –koodin syöttämistä kenttiin ja varmistetaan, etteivät ne voi aiheuttaa haittaa tietokannalle tai ohjelmalle. Lisäksi varmistetaan, että erikoismerkit sekä numerot/kirjaimet sanitoidaan niistä kentistä, joihin ne eivät kuulu, ja annetaan virheilmoitus, mikäli käyttäjä syöttää kelpaamattomia merkkejä kenttiin.

Toteutus

Yritetään syöttää haitallisia php, html ja mysql –koodeja järjestelmään, jolloin on helppoa palauttaa järjestelmä mahdollisista haittatilanteista. Sen lisäksi syötetään erikoismerkkejä ja tarkkaillaan, ettei tietokannasta saa ongittua salattuja salasanoja, eikä tietokannalle pystytä luomaan uusia salasanoja tai poistamaan vanhoja .

Testin vaatimukset

Jos tietokannasta saa poimittua, poistettua, luotua salasanoja tai käyttäjätunnuksia, on testi hylätty. Jos tietokantaan ei pysty vaikuttamaan testin suorittajan kyvyillä, on testi hyväksytty.

Testin lopputulos

Tietokantaan ei pystytty tekemään injektioita, joilla olisi saatu esille, muokattua tai poistettua salasanoja tai käyttäjätunnuksia.

9.3.2 Testitapaus 2: asikastietojen listaus

Käydään ”Näytä kaikki asiakkaat”, ”Näytä tämän kuun tapahtumat”, ”Näytä tulevat tapahtumat”, ”Näytä menneet tapahtumat”, ”Näytä maksamattomat

asiakkaat” ja ”Näytä maksaneet asiakkaat” painikkeet läpi ja verrataan näytölle tulostettavia tietoja tietokannan tietoihin.

Toteutus

Otetaan ajanvarausjärjestelmän testitapaus neljässä luotu dokumentti ja verrataan sitä näytölle tulostuviin tietoihin. Kaikki testitapaus neljän tiedot pitää tulostua ruudulle ryhmittelyjen (painikkeet) valinnan mukaan. Tietokannassa on testitapaus neljän dokumentaation mukaiset tiedot, joten tietokantaa ei ole syytä verrata dokumentaatioon tai näytölle tulostuviin tietoihin, paitsi niissä mahdollisissa tapauksissa, joissa jotkin tiedot jäävät pois valituista kriteereistä.

Testin vaatimukset

Jos tieto, joka on dokumentaatiossa ja tietokannassa ei tulostu näytölle niissä ryhmittelyissä, joihin tieto kuuluu, on testin lopputulos hylätty. Jos taas kaikki dokumentaatiossa ja tietokannassa olevat tiedot tulostuvat näytölle oikeissa ryhmittelyissä, on testi hyväksytty.

Testin lopputulos

Testaus suoritettiin jokaiselle näkymälle useita kertoja. Kokeilukertojen välissä oli lisätty tietoa, poistettu tietoa ja pidetty tieto ennallaan. Mikään näistä ei vaikuttanut näykymiin ja näkymät toimivat odotetuilla tavoin koko testin ajan.

9.3.3 Testitapaus 3: asiakastietojen poistaminen

Testataan tietojen ryhmittelyissä asiakastietojen poistamispainiketta. Poistamispainikkeella pitää sen kohdalla sijaitsevien asiakastietojen poistua tietokannasta, eikä se saa jäädä näkymiin missään ryhmittelyissä. Lisäksi käyttäjän pitää nähdä sekä painaa varmistuspainiketta, jolla varmistetaan, että tiedot todella halutaan poistaa.

Toteutus

Valitaan viisi asiakas tietoa per ryhmittely ja poistetaan ne. Poiston jälkeen verrataan ryhmittelynäkymiä ja varmistetaan tietokannasta, että tieto on varmasti poistunut. Lisäksi tarkkaillaan, että käyttäjä näkee varmistuspainikkeen jokaisella kerralla, ja että varmistuspainike noudattaa siltä odotettuja toimintoja (kyllä poistaa tiedon, älä poista jättää tiedon rauhaan).

Testin vaatimukset

Tietojen pitää aidosti poistua kannasta, samalla poistaen kaikki tiedot, jotka ovat poistettavaan tietoon yhteydessä (relaatio). Lisäksi varmistuspainikkeen pitää toimia sille määritetyllä tavalla (luku 6.3).

Testin lopputulos

Testi suoritettiin toteutustavan mukaisesti, eikä mitään vanhojen tietojen jäänteitä tai niihin liittyviä virheitä ilmennyt tulostusnäkymässä eikä tietokannassa. Testi oli hyväksytty.

9.3.4 Testitapaus 4: uloskirjautuminen

Testataan ”kirjaudu ulos” painiketta ja varmistetaan, että painikkeen painamisen jälkeen asiakas ohjataan rekisterinäkymästä ulos ja kirjautumisen yhteydessä luotu sessiomuuttuja (joka antaa oikeuden selata asiakasrekisteriä) tuhoutuu.

Toteutus

Kirjaudutaan sivustolle sisään sekä ulos 50 kertaa, ja uloskirjautumisten jälkeen yritetään palata asiakasrekisteriin, ilman että uutta sisäänkirjautumista suoritetaan.

Testin vaatimukset

Sessiomuuttujan pitää tuhoutua, ja takaisin asiakasrekisterisivulle pääsyn tulee olla ”tavanomaisin” keinoin mahdotonta ilman uutta sisäänkirjautumista.

Testin lopputulos

Takaisin sivulle yrittäminen ei onnistunut tavanomaisin keinoin, ilman että joutui suorittamaan sisäänkirjautumisen uudestaan. Testi oli hyväksytty.

10 YHTEENVETO

Opinnäytetyö eteni kokonaisuudessaan ongelmitta. Toteutus oli laajuudeltaan yhdelle henkilölle aikaavievä ja se olisi ollut mahdollista toteuttaa nopeammalla aikataululla, jos sitä olisi ollut useampi henkilö toteuttamassa. Koska opinnäytetyössä vaikuttavat henkilöt koostuivat vain tilaajasta ja toteuttajasta, ei ollut tarvetta toteuttaa tarkasti ohjelmistokehityksen kehittämismalleja. Tämä ei olisi ollut mahdollista suuremmalla työmäärällä tai henkilöstöllä, mutta osoittautui tässä toteutuksessa toimivaksi ratkaisuksi kommunikaation helppouden ja hyvän yhteisymmärryksen takia. Kuitenkin tätä suuremmissa projekteissa ja organisaatioissa on tärkeää valita kehittämismalli. Ohjelmistokehityksen näkökulmasta ei ole tärkeää minkä kehittämismallin valitsee, vaan se että yleisesti valitsee kehittämismallin ja seuraa sitä. Olisi kuitenkin hyvä perehtyä erilaisiin kehittämismalleihin ja löytää yritykselle tai työryhmälle parhaiten palveleva malli. Lisäksi tärkeää on valvoa, että mallia noudatetaan. Valitusta kehitysmallista ei kuitenkaan saa tulla itsetarkoitus, vaan sen pitää pystyä palvelemaan toteuttajiaan. Mallin toteuttaminen ei saa muodostua taakaksi. Mallin liiallisuuksiin menevä tarkka valvonta lisää stressiä ja voi hidastaa tai vaikeuttaa projektia. Hyvällä kehitysmallin valinnalla ja toteutuksella pyritään aina nopeuttamaan kehitystä ja vapauttamaan voimavaroja tuotteen luomiseen. Hyvin sisäistetyillä ja opituilla toimintatavoilla on helpompaa tuoda ja kouluttaa uusia ihmisiä osaksi tiimiä.

Testausmenetelmiä on myös lukuisia muita opinnäytetyössä mainittujen lisäksi, mutta esimerkkeinä käytettyjen sekä muiden olemassa olevien tärkein arvo piilee niiden osaavassa toteuttamisessa. Testausmenetelmiä valittaessa olisi hyvä olla riittävä taustatuntemus mahdollisimman monesta eri menetelmästä ja kyky käyttää oikeita menetelmiä tarpeen mukaan. Testausmenetelmät ovat työkaluja, jotka oikein käytettynä myöskin nopeuttavat tuotteen kehitystä sekä varmistavat tuotteen laadun mahdollisimman vähillä resursseilla. Näin pyritään saavuttamaan tavoiteltu lopputulos mahdollisimman tehokkaasti niin ajan kuin resurssien suhteen.

Käytettyjen menetelmien tuomat edut eivät ole kuitenkaan pääasia, vaan tärkeintä on kyky tunnistaa ja toteuttaa tarpeeseen parhaiten sopivat menetelmät niin yrityksen kuin tuotteenkin näkökulmasta.

LÄHTEET

[1] Apache HTTP server project, [www-dokumentti], saatavilla:

http://httpd.apache.org/ABOUT_APACHE.html (Luettu 28.3.2013)

[2] MySQL Licencing Policy, [www-dokumentti], saatavilla:

http://www.mysql.com/about/legal/licensing/index.html

(Luettu 28.3.2013)

[3] InnoDB vs MyISAM vs Falcon Part 1, [www-dokumentti], saatavilla:

http://www.mysqlperformanceblog.com/2007/01/08/innodb-vs-myisam-vs-falcon-benchmarks-part-1/

(Luettu 28.3.2013)

[4] Wikipedia: PHP, [www-dokumentti], saatavilla:

https://en.wikipedia.org/wiki/PHP (Luettu 28.3.2013)

[5] PHP Frameworks , [www-dokumentti], saatavilla:

http://www.phpframeworks.com/index.php (Luettu 28.3.2013)

[6] Advanced Code Injection Techniques and Testing Procedures, [www-dokumentti],saatavilla: http://www.technicalinfo.net/papers/SecondOrderCodeInjection.html (Luettu 28.3.2013)

[7] Secure Software Development, [www-dokumentti], saatavilla:

http://securesoftwaredev.com/security/confidentiality/ (Luettu 28.3.2013)

[8] Wikipedia: Database Normalization > Normal forms , [www-dokumentti], saatavilla: http://en.wikipedia.org/wiki/Database_normalization#Normal_forms(Luettu 28.3.2013)

[9] Tietokannan normalisointi ja normaalimuodot, [PDF-asiakirja], saatavilla:

http://www2.kyamk.fi/~atesa/db/Tietokannat2008_normaalimuodot.pdf (Luettu 28.3.2013)

[10] Luento 4 - ER-mallin muuntaminen relaatiotietokannaksi ja normalisointi, [www-dokumentti],saatavilla: http://appro.mit.jyu.fi/2000/yhteistoiminta/tietokannat/luennot/luento4/ (Luettu 28.3.2013)

[11] Finnish Software Testing Board, [www-dokumentti], saatavilla:

http://www.fistb.fi/ (Luettu 28.3.2013)

[12] White Box Testing Technique, [www-dokumentti], saatavilla:

http://www.codeproject.com/Articles/37111/White-Box-Testing-Technique (Luettu 28.3.2013)

Wikipedia: White Box Testing, [www-dokumentti], saatavilla:

http://en.wikipedia.org/wiki/White-box_testing (Luettu 28.3.2013)

Tutorials Point: Testing >Grey Box Testing, [www-dokumentti], saatavilla:

http://www.tutorialspoint.com/software_testing/testing_methods.htm

(Luettu 28.3.2013)

[13] Testing Overview and Black-Box Testing Techniques, [PDF-asiakirja], saatavila: http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf (Luettu 28.3.2013)

Tutorials Point: Testing > Black Box Testing, [www-dokumentti], saatavilla: http://www.tutorialspoint.com/software_testing/testing_methods.htm

(Luettu 28.3.2013)

[14] Tutorials Point: Testing >Grey Box Testing, [www-dokumentti], saatavilla:

http://www.tutorialspoint.com/software_testing/testing_methods.htm

(Luettu 28.3.2013)

Wikipedia: White Box Testing, [www-dokumentti], saatavilla:

http://en.wikipedia.org/wiki/Gray_box_testing (Luettu 28.3.2013)

Tietokannan relaatiomalli ja tietuekuvaukset

asiakas_tiedot

asiakas_tiedot taulu sisältää kannassa autoinkremetoidun asiakas_id:n, joka toimii primääriavaimena. etunimi, sukunimi, puh_nro ja email-tietueet ovat kaikki itseselitteisiä. etunimi, sukunimi ja email ovat VARCHAR-tyyppisiä ja niiden pituudet näkyvät liitteessä 1, joka on tietokannan relaatiomalli. puh_nro on asiakkaan puhelinnumero ja se on Integer-tyyppinen.

yritys_tiedot

yritys_tiedot taulu on luotu, jotta yritys voi pyytää valokuvaamolta yhteydenottoa, jossa he sopivat tarkemmin tapahtumasta ja hinnoittelusta, koska normaali hinnoittelu vaihtelee yritystilauksissa. yritys_id on autoinkrementoitu primääriavain. yritys_nimi(VARCHAR), yritys_osoite(VARCHAR), yritys_email(VARCHAR) ja yritys_puhro(INT) ovat varmasti yksiselitteisiä (kenttien pituudet ja pakollisuudet löytyvät liite 1:stä). id_asiakas_yritys toimii viiteavaimena asiakas_tiedot tauluun. Näin on muodostettu relaatio, jolla saadaan asiakas/kontaktihenkilö yhdistettyä yritykseen, jonka puolesta hän toimii.

yhtye_tiedot taulu on luotu mahdollisia yhtyeitä varten. Valokuvaamo haluaisi kasvattaa liiketoimintaansa erityisesti tähän suuntaan. yhtye_tiedot koostuu yhtye_id (autoinkrementoitu primääriavain), yhtyeen_nimi(VARCHAR), palvelu_kuvaus(VARCHAR), id_asiakas_yhtye(viiteavain, jolla luodaan relaatio asiakas_tiedot tauluun) tietueista. Palvelu_kuvaus tietue on 1200 merkin mittainen ja sen tarkoituksena on saada lyhyt kuvaus yhtyeen tarpeista. Sen avulla valokuvaamo voi mitoittaa omat resurssinsa jo ennen yhteydenottoa yhtyeeseen.

tapahtuma_tiedot tauluun tallennetaan kaikki tärkeät tiedot koskien varattuja tapahtumia. Taulu koostuu seuraavista tietueista: tapahtuma_id (autoinkrementoitu primääriavain), palvelu ((ENUM) tietueeseen on ennalta

määritelty, mitä arvoja tietue voi saada), tapahtuman_osoite(VARCHAR), tapahtuman_pvm(DATE),tapahtuman_aloitusaika(TIME), tapahtuman_paattymisaika(TIME). tapahtuman_paattymisaika ei ole käytössä ohjelmassa asiakkaan toiveesta, joskin se pystytään ottamaan käyttöön tarvittaessa. varaus_maksettu(BOOLEAN) kertoo valokuvaamolle, onko kyseisestä tapahtumasta maksettu varausmaksu, jolla vahvistetaan tilaus. id_asiakas_tapahtuma on viiteavain jolla luodaan relaatio asiakas_tiedot tauluun.

kastetilaisuus, rippijuhlat, muu, syntymapaivat, haat taulut pitävät sisällään asiakkaan valitsemat palvelut boolean tyypin tietueissa. Tietueissa 1 tarkoittaa valittua palvelua ja 0 ei valittua.

kastetilaisuus taulu koostuu seuraavista tietueista: kastetilaisuus_video(boolean) tai kastetilaisuus_kuvat(BOOLEAN) sekä kastetilaisuus_id (autoinkrementoitu primääriavain) ja id_tapahtuma_kastetilaisuus (viiteavain jolla luodaan relaatio tapahtuma_tiedot tauluun).

rippijuhlat taulu sisältää rippijuhla_id (autoinkrementoitu primääriavain), rippijuhla_video (BOOLEAN), rippi_kuvat (BOOLEAN), rippijuhla_kuvat (BOOLEAN) sekä id_tapahtuma_rippijuhlat (viiteavain, jolla luodaan relaatio tapahtuma_tiedot tauluun) tietueet.

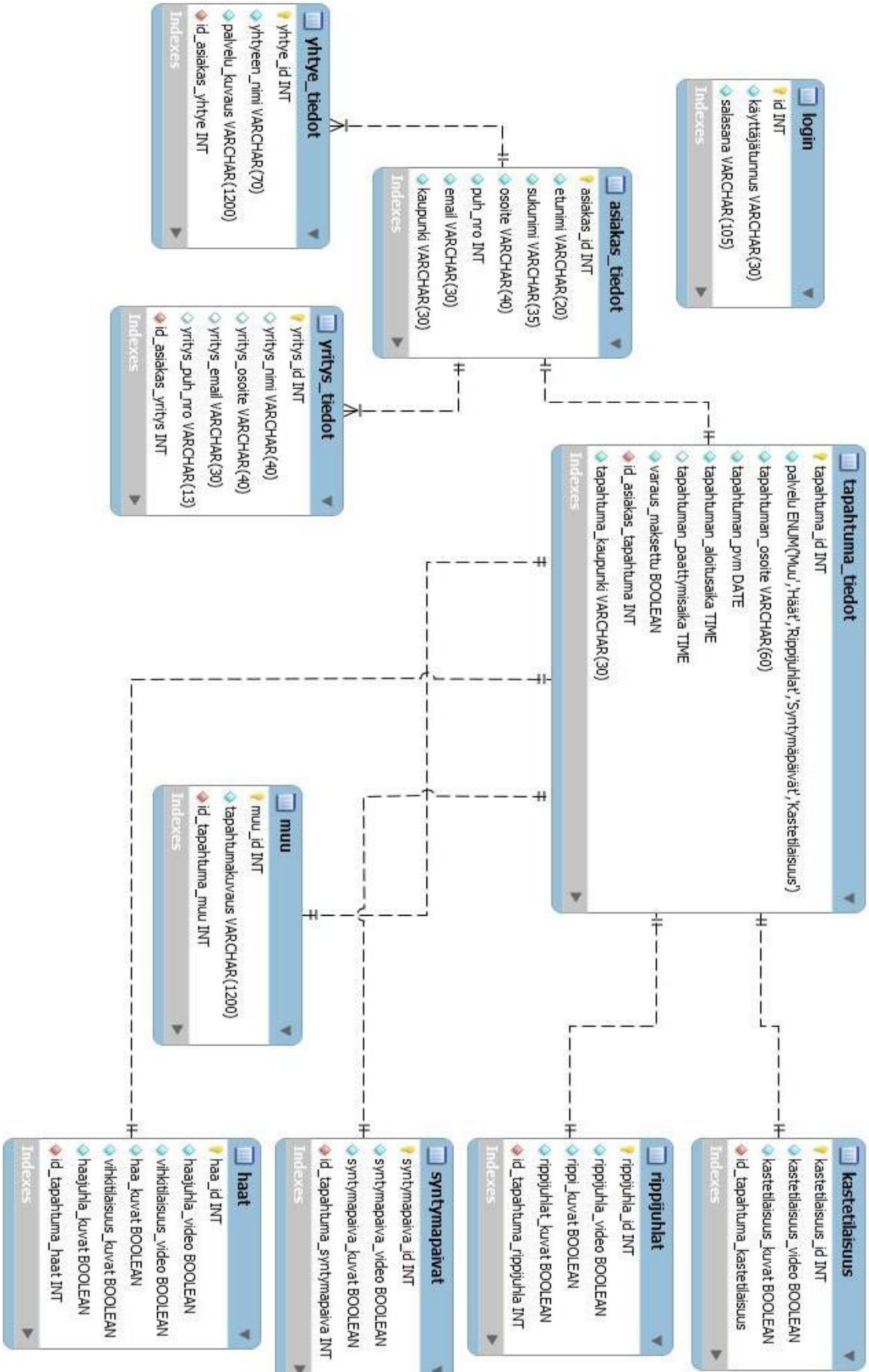
syntymäpäivät taulu sisältää syntymäpäivä_id (autoinkrementoitu primääriavain),s yntymäpäivä_video (BOOLEAN), syntymäpäivä_kuvat (BOOLEAN) sekä id_tapahtuma_syntymäpäivät (viiteavain, jolla luodaan relaatio tapahtuma_tiedot tauluun) tietueet.

muu taulu sisältää muu_id (autoinkrementoitu primääriavain), tapahtuma_kuvaus (VARCHAR) ja id_tapahtuma_muu (viiteavain, jolla luodaan relaatio tapahtuma_tiedot tauluun) tietueet. Taulun tarkoituksena on antaa palvelun tilaajalle mahdollisuus kuvata tarvitsemaansa palvelua lyhyehköllä kuvauksella (1200 merkkiä). Tämä antaa palvelun toimittajalle mahdollisuuden

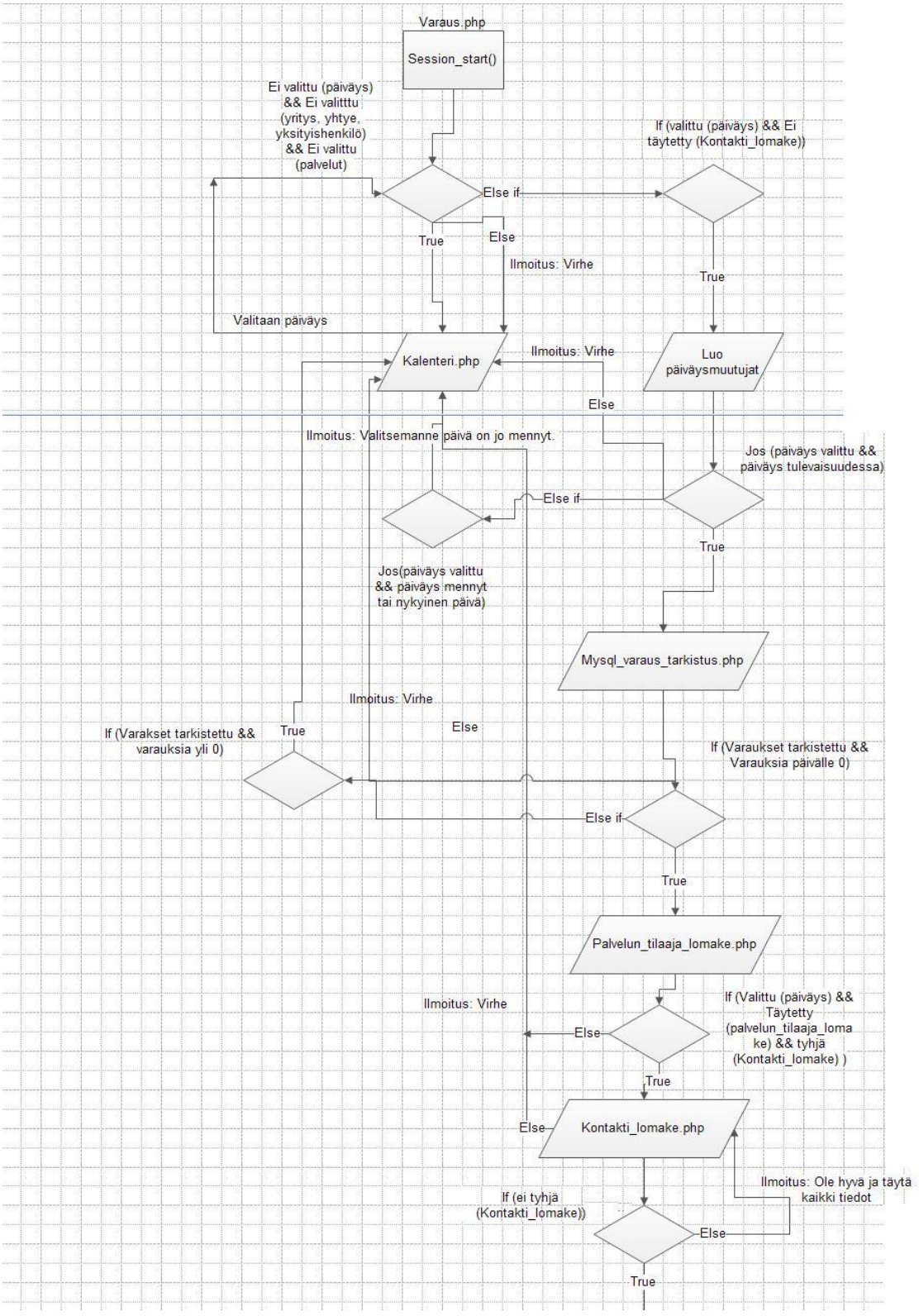
valmistautua ja ottaa tilaajaan yhteyttä tarvittaessa ennen kuin vahvistaa asiakkaan tilauksen.

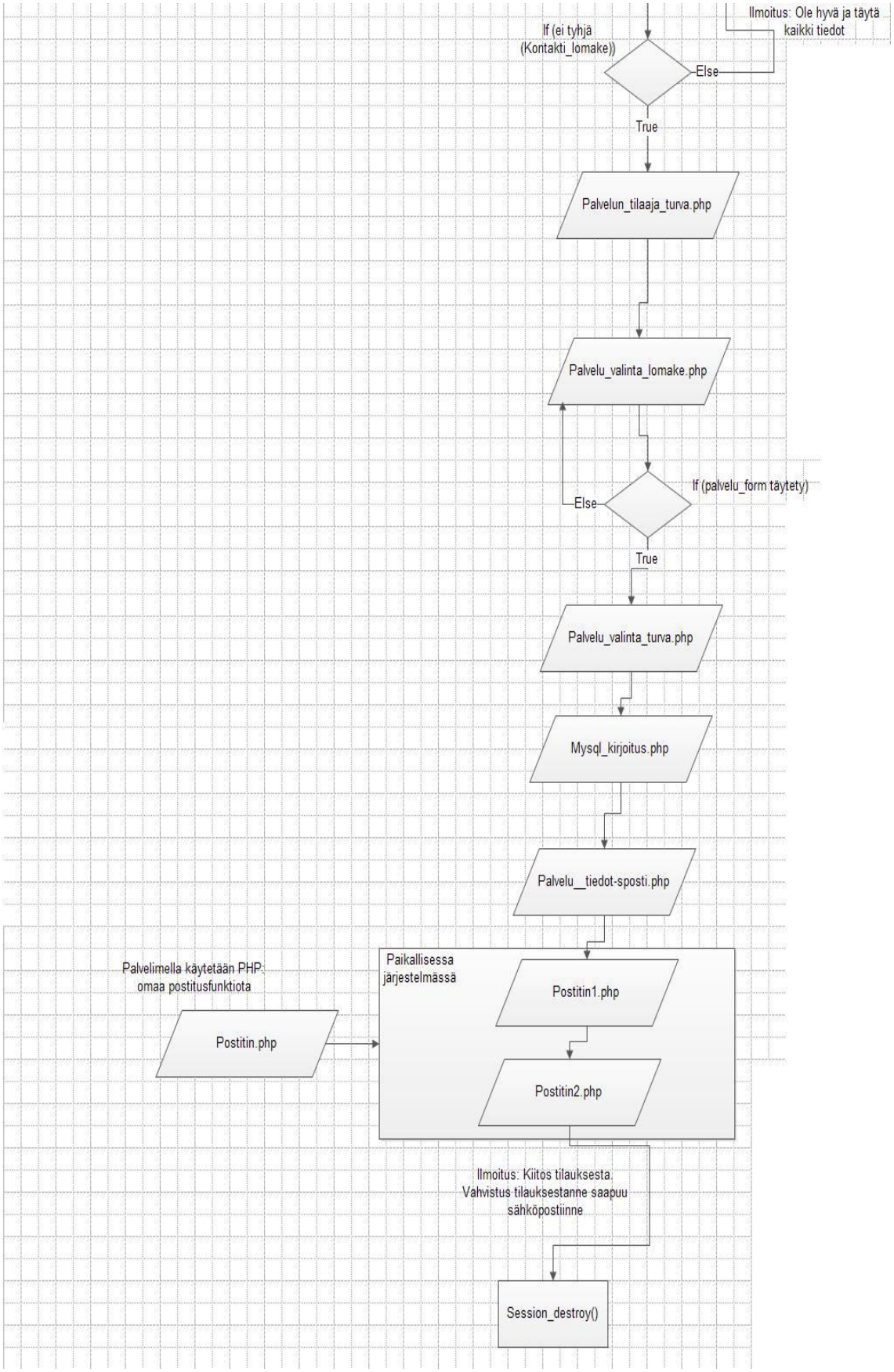
haat taulu koostuu haa_id (autoinkrementoitu primääriavain), haajuhla_video (BOOLEAN), vihkitilaisuus_video (BOOLEAN),haa_kuvat(BOOLEAN), vihkitilaisuus_kuvat (BOOLEAN), haa_juhlakuvat (BOOLEAN) sekä id_tapahtuma_haat tauluista (viiteavain, jolla luodaan relaatio tapahtuma_tiedot tauluun).

Näiden lisäksi tietokannassa on erillään ilman mitään relaatiota login_taulu, jota käytetään sisäänkirjautumisessa asiakasrekisteriin. Taulu koostuu seuraavista tietueista: id (autoinkrementoitu primääriavain), käyttäjätunnus (VARCHAR) ja salasana (VARCHAR). salasana tietueeseen tuleva salasana muodostuu käytetyn salauksen sekä siihen lisätyn suolauksen muunnosyhdistelmästä

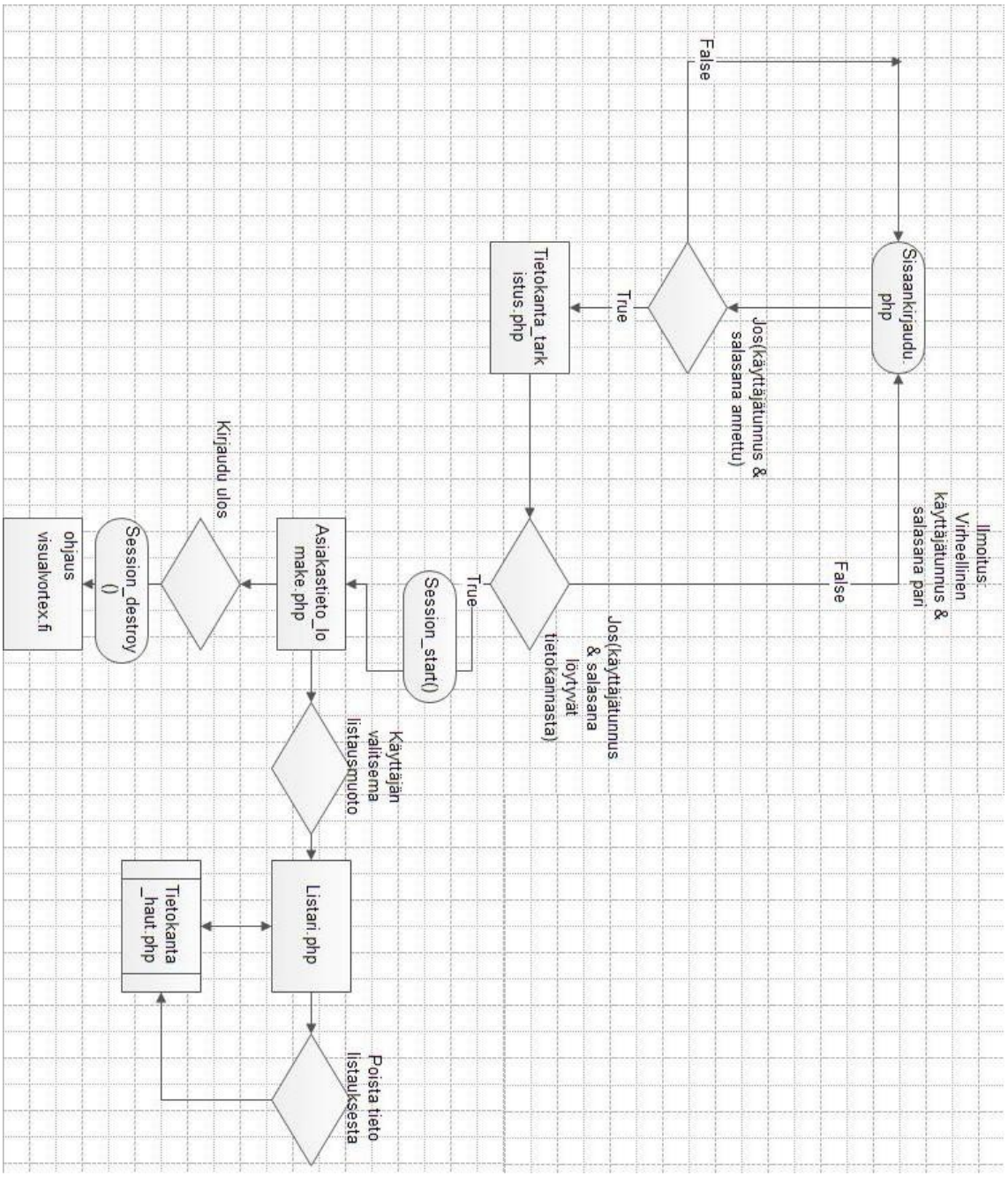

login
id INT
käyttäjätunnus VARCHAR(30)
salasana VARCHAR(105)
Indexes
asiakas_tiedot
asiakas_id INT
etunimi VARCHAR(20)
sukunimi VARCHAR(35)
osoite VARCHAR(40)
puh_nro INT
email VARCHAR(30)
kaupunki VARCHAR(30)
Indexes
yhtye_tiedot
yhtye_id INT
yhtyeen_nimi VARCHAR(70)
palvelu_kuvaus VARCHAR(1200)
id_asiakas_yhtye INT
Indexes
yritys_tiedot
yritys_id INT
yritys_nimi VARCHAR(40)
yritys_osoite VARCHAR(40)
yritys_email VARCHAR(30)
yritys_puh_nro VARCHAR(13)
id_asiakas_yritys INT
Indexes
tapahtuma_tiedot
tapahtuma_id INT
palvelu ENUM('Muu','Häät','Rippijuhlat','Syntymäpäivät','Kastetilaisuus')
tapahtuman_osoite VARCHAR(60)
tapahtuman_pvm DATE
tapahtuman_aloitusaika TIME
tapahtuman_paattymisaika TIME
varaus_maksettu BOOLEAN
id_asiakas_tapahtuma INT
tapahtuma_kaupunki VARCHAR(30)
Indexes
kastetilaisuus
kastetilaisuus_id INT
kastetilaisuus_video BOOLEAN
kastetilaisuus_kuvat BOOLEAN
id_tapahtuma_kastetilaisuus
Indexes
rippijuhlat
rippijuhla_id INT
rippijuhla_video BOOLEAN
rippi_kuvat BOOLEAN
rippijuhlat_kuvat BOOLEAN
id_tapahtuma_rippijuhla INT
Indexes
muu
muu_id INT
tapahtumakuvaus VARCHAR(1200)
id_tapahtuma_muu INT
Indexes
syntymapaivat
syntymapaiva_id INT
syntymapaiva_video BOOLEAN
syntymapaiva_kuvat BOOLEAN
id_tapahtuma_syntymapaiva INT
Indexes
haat
haa_id INT
haajuhla_video BOOLEAN
vihkitilaisuus_video BOOLEAN
haa_kuvat BOOLEAN
vihkitilaisuus_kuvat BOOLEAN
haajuhla_kuvat BOOLEAN
id_tapahtuma_haat INT
Indexes

Ohjelman prosessikaavio: Ajanvarausjärjestelmä

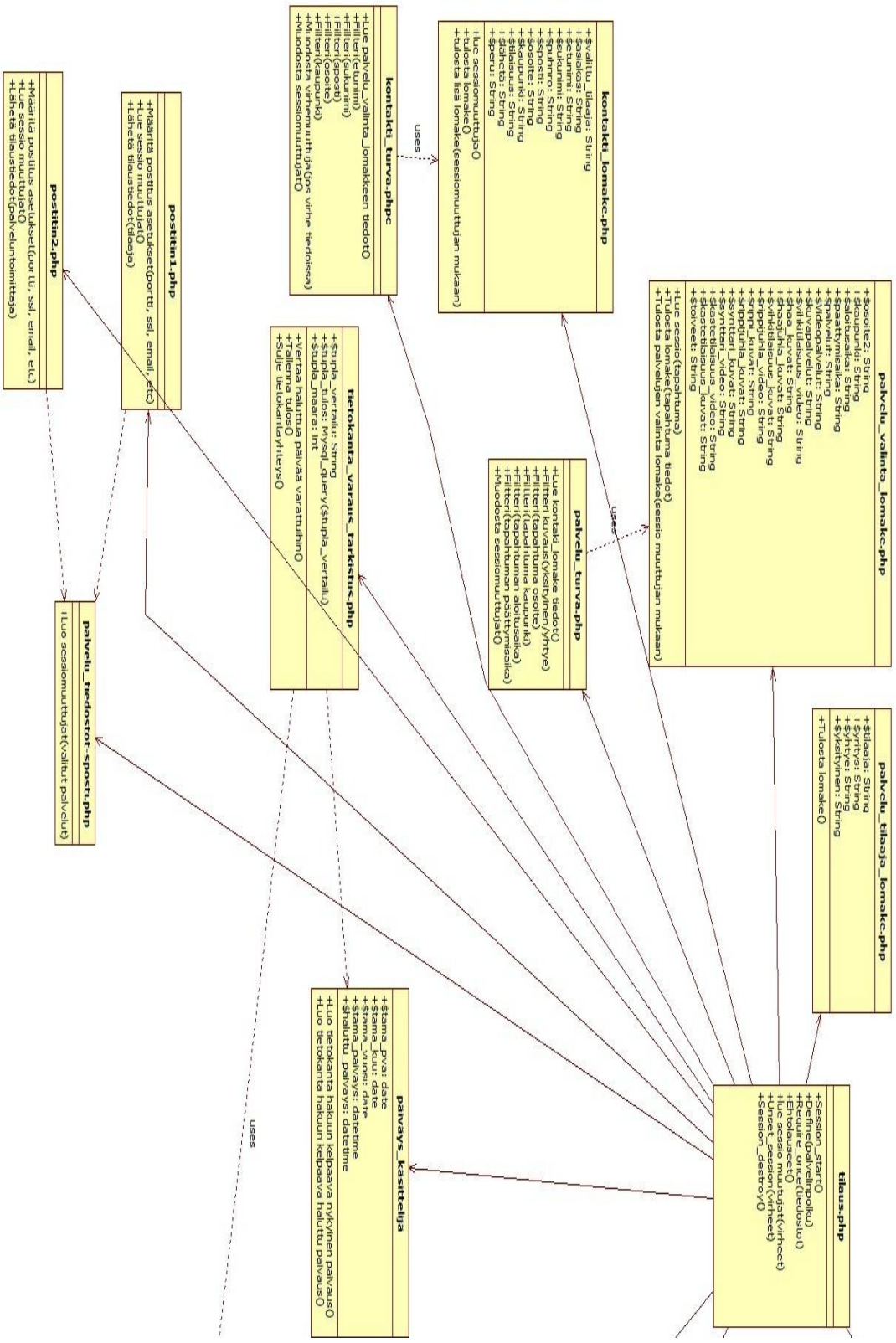


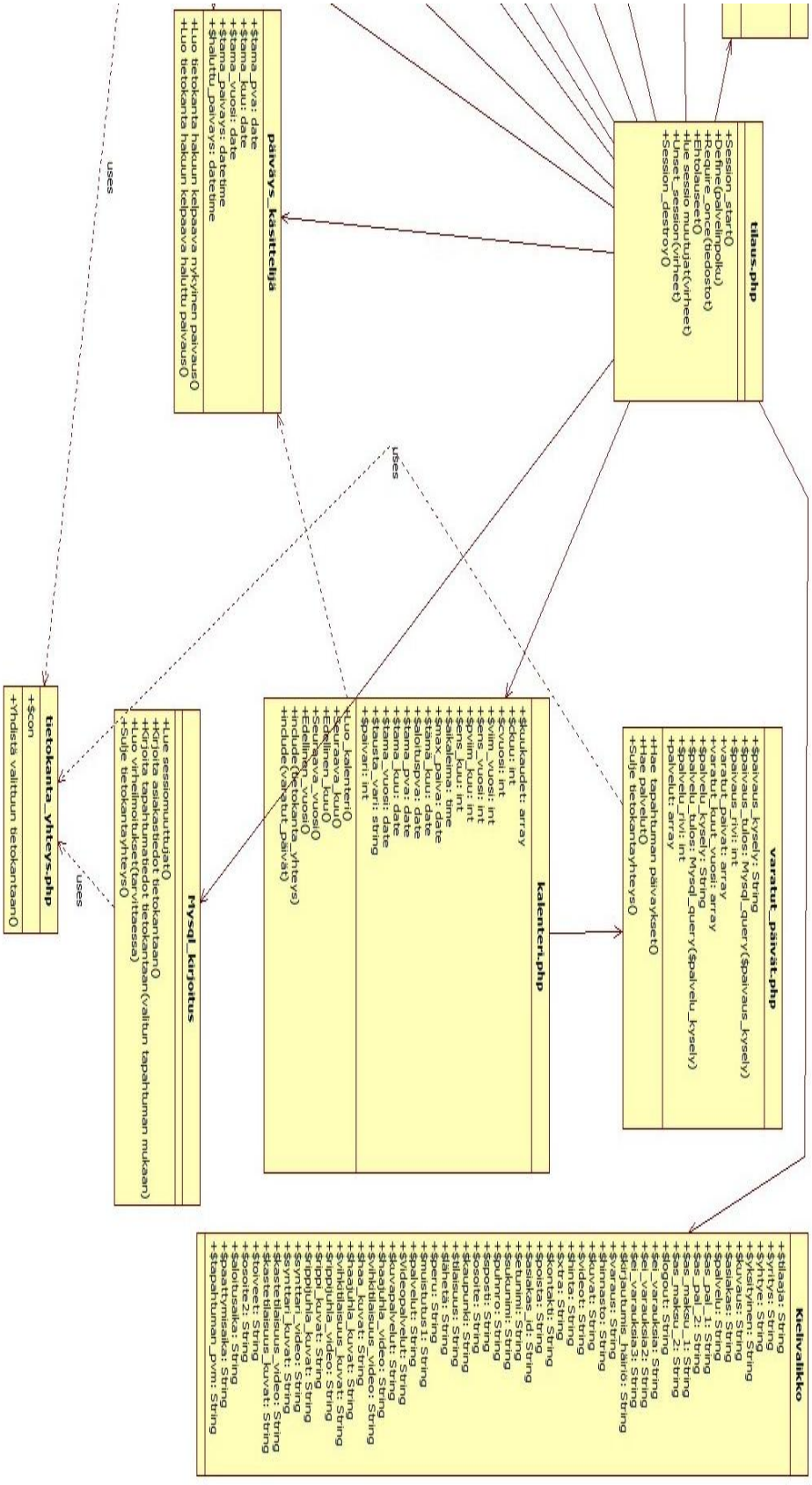

Ilmoitus: Ole hyvä ja täytä kaikki tiedot
If (ei tyhjä (Kontakti_lomake))
Else
True
Palvelun_tilaaja_turva.php
Palvelu_valinta_lomake.php
If (palvelu_form täytety)
Else
True
Palvelu_valinta_turva.php
Mysql_kirjoitus.php
Palvelu__tiedot-sposti.php
Palvelimella käytetään PHP: omaa postitusfunktiota
Postitin.php
Paikallisessa järjestelmässä
Postitin1.php
Postitin2.php
Ilmoitus: Kiitos tilauksesta. Vahvistus tilauksestanne saapuu sähköpostiinne
Session_destroy()

Ohjelman prosessikaavio: Asiakasrekisteri



Ohjelman luokkadiagrammi: Kalenteri




tilaus.php
+Session_start()
+Define(palvelinpolku)
+Require_once(tiedostot)
+Istuntomuuttujat()
+Lue sessio muutujat(virheet)
+Unset_session(virheet)
+Session_destroy()
päivays_kasittelija
+$tama_pva: date
+$tama_kuu: date
+$tama_vuosi: date
+$tama_paivays: datetime
+$haluttu_paivays: datetime
+Luo tietokanta hakuun kelpaava nykyinen paivaus()
+Luo tietokanta hakuun kelpaava haluttu paivaus()
uses
uses
uses
tietokanta_yhteys.php
+$con
+Yhdistä valittuun tietokantaan()
Mysql_kirjoitus
+Lue sessiomuuttujat()
+Kirjoita tietokantaan(tietokantaan)
+Kirjoita tapahtumatiedot tietokantaan(valitun tapahtuman mukaan)
+Luo virhelmoitukset(tarvittaessa)
+Sulje tietokantayhteys()
kalenteri.php
+$tukukaudet: array
+$kuu: int
+$vuosi: int
+$viim_vuosi: int
+$ens_vuosi: int
+$viim_kuu: int
+$ens_kuu: int
+$aikaleima: time
+$nykypaiva: date
+$tama_kuu: date
+$aloituspva: date
+$tama_pva: date
+$tama_kuu: date
+$tama_vuosi: date
+$tausta_vari: string
+$paivari: int
+Luo_kalenteri()
+Seuraava_kuu()
+Edellinen_kuu()
+Seuraava_vuosi()
+Edellinen_vuosi()
+include(tietokanta_yhteys)
+include(valatut_paivat)
varatut_paivat.php
+$paivaus_kysely: String
+$paivaus_tulos: Mysql_query($paivaus_kysely)
+$paivaus_rivi: int
+varatut_paivat: array
+varatut_kuut_vuosi: array
+$palvelu_kysely: String
+$palvelu_tulos: Mysql_query($palvelu_kysely)
+$palvelu_rivi: int
+palvelut: array
+Hae tapahtuman päivaykset()
+Hae palvelut()
+Sulje tietokantayhteys()
kielivalikko
+$tilaaja: String
+$ytunnus: String
+$yritys: String
+$yksityinen: String
+$kuvaus: String
+$asiakas: String
+$palvelu: String
+$as_pal_1: String
+$as_pal_2: String
+$as_maksu_1: String
+$as_maksu_2: String
+$logout: String
+$ei_varauksia: String
+$ei_varauksia2: String
+$ei_varauksia3: String
+$varaus: String
+$nimesto: String
+$kuvat: String
+$videot: String
+$hinta: String
+$extra: String
+$poista: String
+$asiakas_id: String
+$etunimi: String
+$sukunimi: String
+$puhnro: String
+$sposti: String
+$kaupunki: String
+$tilaisuus: String
+$tiheta: String
+$peru: String
+$muistutus1: String
+$muistutus2: String
+$videopalvelut: String
+$kuvapalvelut: String
+$haajuhla_video: String
+$vihkitilaisuus_video: String
+$haa_kuvat: String
+$haajuhla_kuvat: String
+$vihkitilaisuus_kuvat: String
+$ripplihla_video: String
+$ripplihla_kuvat: String
+$ripp_kuvat: String
+$ripplihla_kuvat: String
+$synttari_video: String
+$synttari_kuvat: String
+$kastetilaisuus_video: String
+$kastetilaisuus_kuvat: String
+$tavoite1: String
+$tavoite2: String
+$sosoite2: String
+$aloitusaika: String
+$paattymisaika: String
+$tapahtuman_pvm: String

Ohjelman luokkadiagrammi: Ajanvarausjärjestelmä