# SÄHKÖISEN KANTA-ASIAKASJÄRJESTEL-MÄN KEHITYSTYÖ

**Jalmari Lehto**

**Anton Vesterinen**

**Opinnäytetyö**

**Toukokuu 2008**

**Liiketalous**

JYVÄSKYLÄN
AMMATTIKORKEAKOULU

JYVÄSKYLÄN
AMMATTIKORKEAKOULU

KUVAILULEHTI
Päivämäärä
15.7.2008

Tekijä(t)
LEHTO, Jalmari

VESTERINEN, Anton

Julkaisun laji
Opinnäytetyö

Sivumäärä
73

Julkaisun kieli
suomi

Luottamuksellisuus
☐ Salainen _____saakka

Työn nimi
SÄHKÖISEN KANTA-ASIAKASJÄRJESTELMÄN KEHITYSTYÖ

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Työn ohjaaja(t)
BISTER, Timo

Toimeksiantaja(t)
-

Tiivistelmä

Työn tavoite oli luoda sähköinen kanta-asiakasjärjestelmä ravintola-alalle omien ideoiden pohjalta. Tarkoitus oli raportoida kehitystyön kulku ideatasolta valmiiksi tietojärjestelmäksi. Useat tietojärjestelmät saavat alkunsa nimenomaan tarpeista ja ideoista niiden tyydyttämiseksi. Tämä kehitystyö kertoo miten tietojärjestelmän voi toteuttaa tuotteeksi asti ja mitä asioita on otettava huomioon järjestelmää kehitettäessä.

Kehitystyö toteutettiin suunnittelemalla järjestelmän toiminnallisuudet käyttäen hyväksi asiakashallintajärjestelmien teoriaa ja yleisiä toimintaperiaatteita. Tutkimustietoa asiakkuudenhallintajärjestelmistä saatiin lähinnä nettiartikkeleista. CRM (Customer Relationship Management) nimen alle on luotu valtava määrä erilaisia asiakkuudenhallintajärjestelmiä, ja niiden perusperiaatteita noudattaen oli tarkoitus kehittää nimenomaan ravintola-alalle suunnattu järjestelmä.

Järjestelmän tekniseen toteutukseen käytettiin runsaasti aikaa ja tästä vaiheesta pyrittiin raportoimaan tarkasti. Tarkoitus oli antaa lukijalle käsitys siitä miten omat ideat ja tärkeät toiminnallisuudet saadaan ensin toimivaksi järjestelmäksi, ja miten valmis järjestelmä saadaan julkiseksi tuotteeksi, jolla on kehityskaari ja laajenemismahdollisuudet. Teknisissä kysymyksissä ja ongelmissa pyrittiin hyödyntämään myös alan lähdekirjallisuutta ja internetiä.

Työn tuloksena syntynyt tietojärjestelmä on esimerkki siitä, miten oma järjestelmä voidaan toteuttaa. Kokemuksen puutteen vuoksi ongelmiakin esiintyi ja uusia asioita kokeiltiin ja opittiin. Yleisistä tietojärjestelmäkehityksen periaatteista haluttiin korostaa suunnittelun ja dokumentoinnin vaikutusta järjestelmän jatkokehitykseen. Järjestelmän tulevaisuutta on hyvä miettiä jo ensimmäistä versiota kehitettäessä. Tietojärjestelmä on jatkuvasti kehittyvä tuote.

Avainsanat (asiasanat)

tietojärjestelmät, asiakkuudenhallinta, www-sivustot

Muut tiedot

JYVÄSKYLÄ UNIVERSITY OF APPLIED SCIENCES

DESCRIPTION
Date
15/07/2008

Author(s)
LEHTO, Jalmari

VESTERINEN, Anton

Type of Publication
Bachelor´s Thesis

Pages
73

Language
finnish

Confidential
☐ Until_____

Title
DEVELOPMENT OF ELECTRONIC CRM-SYSTEM

Degree Programme
Degree Programme in Business Information Systems

Tutor(s)
BISTER, Timo

Assigned by
-

Abstract

The main objective of this thesis was to create a web-based CRM-system for night club industry based on researchers own ideas. Progress of the development was reported from ideas to complete information system and commercial product. Many information systems begin with needs and thoughts to satisfy those needs. This development shows how information system evolves to commercial product and what are the main issues to pay attention to.

Development was accomplished by designing the features using basic knowledge and principles from CRM-field. Source material from that particular field was mainly found from internet articles. There are many different systems built under the term CRM and intention was to make one specifically for the night clubs.

Large amount of time was spent on the technical implementation and this particular stage was reported in detail. Reader is given information how those own ideas and important features are first turned in to operational system, and then, how the system is turned in to public commercial product that has potential for extensions and further development. Source material from IT-industry and internet was used to solve technical problems and questions.

CRM-system, as result from development, is an example, how one can turn own ideas into information system. Some problems were met too, due to lack of experience and new methods were tested and learnt. The benefits of detailed designing and documenting come forth when development of the system is continued. Future development must be considered already when developing the first version of program. Information system is forever evolving product.

Keywords
CRM, information systems, websites

Miscellaneous

# SISÄLTÖ

# KUVIOT

# 1 JOHDANTO

Kaikilla yrityksillä on hallinnoitavanaan erilaista tietoa. Tiedon määrä vaihtelee yrityksen koon ja toimialan mukaan. Monet yritykset eivät itse ymmärrä omistamansa tiedon määrää. Tiedon systemaattinen hallinta parantaa yrityksen käsitystä omasta toiminnastaan ja edistää liiketoimintaa monin eri tavoin. Yleensä tärkeintä tietoa yrityksessä ovat asiakastiedot. Ei ole syytä väheksyä myöskään yhteistyökumppaneihin ja henkilökuntaan liittyvää tietoa. Näiden tietojen varastointiin on viimevuosina alettu kehittää toinen toistaan toimivampia ja tehokkaampia järjestelmiä.

Tämä opinnäytetyö on täysin tyhjästä aloitettu tuotteen kehitys, elektroninen kanta-asiakasjärjestelmä, joka on suunnattu erityisesti yökerhojen, miksei ravintoloidenkin käyttöön. Järjestelmä tulee hyödyntämään internetiä ja laajakaistojen leviämistä, joten kanta-asiakkaiden mielenkiintoa sidotaan yökerhoon silloinkin, kun asiakkaat eivät ole paikan päällä. Asiakaspalvelua voidaan näin hoitaa ravintola-ajan ulkopuolella.

Tämän järjestelmän kehittäjille projekti saattaa olla merkittävämpi kuin järjestelmän tuleville asiakkaille, koska tämä on tekijöiden ensimmäinen suurempi kehitysprojekti ja antaa näin korvaamatonta kokemusta alalta. Tarkoitus on ilmentää, kuinka yritysmaailman tarpeisiin voidaan luoda nykytekniikkaa hyödyntäviä sähköisiä työkaluja ideointitasolta valmiiksi tuotteeksi asti.

# 2 TUTKIMUSASETELMA

## 2.1Opinnäytetyön taustateoriaa, tavoitteet ja rajaukset

Markkinointia on jo pitemmän aikaa hoidettu useissa alan yrityksissä internetin ja sähköpostin välityksellä, mutta tämän mainonnan yhdistäminen järjestelmään, joka tunnistaa asiakkaiden käyttäytymisen ja mahdollisesti lisää sitä sekä luo lisäksi vahvemman imagon yökerholle, nostaa markkinoinnin tehokkuuden korkeampaan potenssiin.

Tällaiset kanta-asiakasjärjestelmät ovat alalla toistaiseksi harvinaisia (perustuen siihen mitä oma tietoutemme, kuulopuheet ja internetistä haettujen osumien määrä ja laatu kertoo) ja niitä tarjoavat lähinnä pienet paikalliset alan yritykset, niinkuin meidänkin tapauksessamme. Tietysti myös suurilla tuottajilla on tuotteensa, mutta niiden hintataso ylittää yksityisten ja pienten ketjujen taloudelliset rajat. Laajempi leviäminen yökerhoalalle on varmasti tulossa.

Kun lähdetään kehittämään järjestelmää, joka toimii maailmanlaajuisessa verkossa ja periaatteessa kaikilla ihmisillä on mahdollisuus päästä käyttämään järjestelmää, tulevat jo suunnitteluvaiheessa eteen tietoturvakysymykset. Niihin on perehdyttävä ja selvimmin ne vaikuttavat valittaessa tekniikkaa, millä järjestelmä toteutetaan. Tietoturvan määrä on kuitenkin suhteutettava järjestelmän sisältämän informaation määrään ja sitä kautta järjestelmän kustannuksiin, joten kompromissejakin on varmasti tehtävä.

Asiakkuuden hallinnasta ja siihen liittyvien apuvälineiden käytöstä kerrotaan oleellisimmat tiedot ja periaatteet. Tietojärjestelmän kehitystyö ja toteuttaminen olivat kuitenkin projektin suurin ja työläin osuus, ja tästä vaiheesta pyritään raportoimaan tarkemmin. Työn teknisiä näkökulmia on helpompi lähestyä, jos perehtyy ensin yleisimpiin web-ohjelmointitekniikoihin ja niiden perusteisiin. Tämän työn tarkoituksena on mennä syvemmälle web-pohjaisen tietojärjestelmän toteutustapoihin ja tekniikoihin sekä antaa esimerkki siitä, kuinka webhosting-palvelua hyödyntäen voidaan toteuttaa julkinen tietojärjestelmä.

Tehtävänä ja tavoitteena on siis luoda järjestelmä, joka on riittävän tietoturvallinen ja josta asiakas eli yökerho hyötyisi mahdollisimman paljon. Rajaamme työmme aiheen

asiakkuudenhallintajärjestelmiin ja tietojärjestelmän kehitystyöhön. Jätämme tuotteistamiseen ja tuotteen kaupallisuuteen liittyvät asiat oman yrityksemme selvitettäväksi tulevaisuudessa.

## 2.2Tutkimusmenetelmät

Tutkimus toteutetaan pääosin kehittämällä ja toteuttamalla omien ideoiden pohjalta luotu asiakashallintajärjestelmä ravintola-alalle sekä tukemalla opittuja asioita kirjallisella lähdemateriaalilla. Tutkimuksen metodina on alkuasetelmista ja tavoitteista johtuen kehittämistutkimus. Kun tarkoituksena on lisäksi luoda jotain uutta ideoiden pohjalta on kyseessä osittain myös konstruktoiva tutkimus.

## 2.3Tutkimuskysymykset

Järjestelmää kehitettäessä pyritään ottamaan huomioon työn kaksi keskeistä kysymystä. Ensimmäinen kysymys kuuluu, mitä vaatimuksia järjestelmän käyttäjät asettavat sähköiselle kanta-asiakas järjestelmälle? Järjestelmän käyttäjällä tarkoitetaan ravintolan tai yökerhon johtoa ja henkilökuntaa, jotka hallinnoivat järjestelmää ja hyötyvät sen antamista tiedoista, sekä ravintolan tai yökerhon kanta-asiakkaita, jotka käyttävät järjestelmää asiakkaana. Teknisempi kysymys on, mitä tietojärjestelmien kehitykseen vaikuttavia asioita tulee ottaa huomioon järjestelmää toteutettaessa? Tähän kysymykseen pyritään vastaamaan niin teoreettisella tasolla, kuin käytännössäkin.

# 3 ASIAKKUUDEN HALLINTA JA CRM

## 3.1Asiakkuuden hallinta käsitteenä

Asiakkuuden hallintaa, josta käytetään myös termiä asiakkuusajattelu, on käsitteenä hankala kuvata. Sille ei voi antaa kovinkaan tarkkaa lyhyttä selitystä, koska siihen sisältyy niin paljon faktoja, olosuhdevaihteluita ja mielipiteitä. Yleistäen voidaan kuitenkin sanoa, että asiakkuuden hallinnalla tarkoitetaan yrityksen asiakkaisiin liittyvien tietojen keräämistä, hallinnointia ja hyväksi käyttämistä. Asiakkuuden hallinnan tulisi mahdollistaa asiakkuuden elinkaari aina asiakkaan löytämisestä asiakkaan pitämiseen, ja mahdollisesti myös asiakassuhteen vahvistumiseen. (Kaskela 2005)

Asiakkuusajattelu voi tarkoittaa eri yrityksille eri asioita. Eri alojen yritykset tarvitsevat asiakkaistaan erilaisia tietoja. Perustieto asiakkaasta on kuitenkin yleensä sama, asiakkaan henkilötiedot. Lisäksi halutaan seurata asiakaskäyttäytymistä, jonka perusteella pystytään tunnistamaan esimerkiksi yrityksen keskeinen asiakaskunta ja parhaat asiakkaat. Tätä kautta kyetään kohdistamaan markkinointia ja myyntiä, mikä taas tähtää asiakasuskollisuuteen. Asiakkuuden hallintaa helpotetaan erilaisilla tietojärjestelmillä ja ohjelmilla, joista käytetään usein nimitystä CRM-järjestelmä. (Kaskela 2005)

## 3.2Mitä CRM tarkoittaa?

Asiakkuuden hallintaan soveltuvaa järjestelmää etsivän kannattaa kokeilla internetissä hakusanana lyhennettä CRM, jolloin törmää niin suomalaisiin kuin ulkomaalaisiinkin asiakkuudenhallintaohjelmistoihin. Yleensä CRM on lyhenne sanoista Customer Relationship Management, asiakassuhteen hallinnointi. Kyseisen kirjainlyhenteen merkitystä ja siihen liittyvien järjestelmien käyttöaluetta on kuitenkin alettu laajentaa viime vuosina. Uusi termi koostuu englanninkielisistä sanoista Constituent Relationship Management. Customer, eli asiakas, on korvattu sanalla Constituent, joka viittaa kaikkiin henkilöihin ja tahoihin, jotka ovat yrityksen kanssa tekemisissä. Tähän ryhmään voidaan asiakkaiden lisäksi laskea muun muassa henkilökunta, rahoittajat, yhteistyökumppanit, media (lehdet, televisio, radio), kilpailijat ja alihankkijat. Suomenkielinen

olevia menetelmiä ja teknisiä apuvälineitä, joilla pyritään hankkimaan, pitämään ja vahvistamaan suhteita näihin osallisiin. Tässä työssä kirjaimilla CRM viitataan kuitenkin lähinnä asiakkuuden hallintaan. (Hagen 2006)

## 3.3CRM:n pääperiaatteet ja hyöty yritykselle

Ajateltakoon CRM:ää sitten vain asiakkaihin liittyvänä tai kaikkiin osallisiin liittyvänä, on sen toteuttaminen yrityksissä yksilöllistä. CRM:n tarve ja sen käyttölaajuus riippuu usein yrityksen koosta ja toimialasta sekä mahdollisesti yrityksen toimintaperiaatteista. Parikymmentä vuotta sitten elettiin aikaa, jolloin kauppias tunsi suurimman osan asiakkaistaan nimeltä ja tiesi jo ennalta, mitä tuotteita he aikoivat ostaa. Tällaisia asiakassuhteita ei juuri synny nykyisin. Syitä tähän on monia. Asiakkaiden ja kauppojen määrät ovat moninkertaistuneet vuosien saatossa, ihmiset käyttävät useampia ostospaikkoja ja henkilökunta vaihtuu tiheään. Uudet CRM-ratkaisut pyrkivät kuitenkin tavallaan palauttamaan osan tästä asiakaskulttuurista. Sen sijaan, että kauppiaan tai kassatyöntekijän tarvitsisi muistaa, kuka kukin on, ja mitä he yleensä ostavat, tietojärjestelmät hoitavat asian heidän puolestaan. Nykyään ihmiset saavat tarjoukset ja tiedotteet omasta kantakaupastaan lehtimainoksena, tekstiviestinä tai sähköpostitse. Sen sijaan, että tuttu kauppias antaa alennusta tutulle asiakkaalleen, saa asiakas tänä päivänä bonusta ostoistaan. Tämän bonuksen on hänelle laskenut CRM-järjestelmä. Christian Sarkar painottaa CRM-järjestelmien tulemiseen liittyvässä artikkelissaan, että nykyään pyritään markkinaosuuden sijaan kasvattamaan nimenomaan asiakkaiden määrää. Markkinaosuuden kasvaminen syntyy täten kasvaneen asiakasvolyymin kautta. (Hagen 2006, Meltzer 2004)

Tehokkaan asiakkuuden ja osallisten hallinnan etuja on alettu miettiä tarkemmin ja laajemmin viime vuosien aikana. Edelleen toimii paljon yrityksiä, varsinkin pieniä, jotka eivät ymmärrä tehostetun asiakkuuden hallinnan vaikutusta suoraan yrityksen tulokseen. Asiakkuuden hallintaa on montaa eri tasoa ja tyyliä. Paul Hagenin vuoden 2006 artikkeli ’Creating the Relationship-Centric Organization: Nonprofit CRM’ keskittyy enimmäkseen sellaisten yhdistysten ja seurojen toimintaan, jotka eivät tavoittele taloudellista voittoa. Kyseisessä artikkelissa on kuitenkin hyvin yleispätevä kaavio (kuvio 1) siitä, millaisia koulukuntia yritykset edustavat liittyen heidän tapaansa säilyttää tietoa osallisistaan. (Hagen 2006)


KUVIO 1. CRM:n eri tasot. (Hagen, 2006)

Nykyaikainen CRM perustuu tietojen keskittämiseen. Edellisen vuosituhannen lopussa elettiin vielä enimmäkseen Microsoft Office -ajassa, jossa yrityksen tietoja tallennettiin sinne tänne erillisiin Excel-tauluihin tai Access-tietokantoihin. Uusi vuosituhat on tuonut mukanaan keskitetyn tiedonhallinnan. Yrityksen tiedot pyritään sisällyttämään yhteen, korkeintaan muutamaan, erilliseen tietopankkiin. Tietojen hallintaa varten on kehitetty tarkoituksenmukaisia hallintaohjelmistoja, jotka helpottavat suuresti tietojen käsittelyä verrattuna Office-aikaan. Tietojen keskittämisen kautta pystytään analysoimaan asiakastietoa niin, että yhteydenottojen laatua kyetään parantamaan. Yhteydenotto tarkoittaa tässä yhteydessä mitä tahansa kontaktia asiakkaaseen tai osalliseen, esimerkiksi myyntitapahtumaa, tarjouspyyntöä, tarjouksen tekemistä, tyytyväisyyskyselyä tai uuden työntekijän palkkaamista. Lähes poikkeuksetta asiakkuuden hallintaan tarkoitettua ohjelmistoa tai järjestelmää käyttää ihminen. Verrattuna siihen, että asiakastiedot tulisi etsiä ja yhdistellä useista eri tietolähteistä (esimerkiksi Excel-tiedostoista ja arkistopapereista), on selvää, että työn tekeminen helpottuu ja tehostuu keskitetyn tiedonhallinnan ansiosta. (Hagen 2006, Meltzer 2004)

Pienen yrityksen CRM:n tarve voi olla vain asiakkaiden perustietojen hallinnassa ja laskutuksessa. Jos työntekijöitä on enemmän kuin muutama, voi olla järkevää sähköistää myös palkanlaskenta. Tähän tarkoitukseen on olemassa edullisia ratkaisuja, esimerkiksi televisiosta tutun Passeli-ohjelman halvemmat versiot. Suuret yhtiöt tarvitsevat monesti monimutkaisempia ratkaisuja, jolloin valmiiden ohjelmien soveltuvuus ei ole enää riittävä. Tähän tarkoitukseen on olemassa yrityksiä, jotka räätälöivät halutunlaisen CRM-kokonaisuuden. Tällaisten järjestelmien toteuttaminen vie huomattavasti enemmän rahaa kuin valmiit ratkaisut, joten pienillä yrityksillä on harvemmin tarvetta ja varaa ostaa räätälöityjä CRM-järjestelmiä. (Hagen 2006)

# 4 JÄRJESTELMÄN LÄHTÖKOHDAT, TAVOITTEET JA TOTEUTUSTEKNIIKAT

## 4.1Lähtökohdat

Opinnäytteen käytännön osa on kanta-asiakasjärjestelmän kehitystyö. Idea sai alkunsa, kun toinen työn tekijöistä oli työharjoittelussaan tradenomin tutkintoon kehittänyt ja ohjelmoinut alkeellisen asiakaskäyntejä laskevan korttijärjestelmän tanssiravintolan käyttöön. Hänen työparinsa ehdotti harjoittelun jälkeen, että tuota ideaa hyödynnettäisiin ja laajennettaisiin ja kehitettäisiin oma, huomattavasti edistyksellisempi järjestelmä nimenomaan ravintola-alaa ajatellen. Suurin osa järjestelmän ominaisuuksista syntyi puhtaalta ideointipohjalta, maalaisjärjellä. Myöhemmin ideoita jalostettiin niin, että otettiin huomioon tärkeimpiä CRM-ohjelmistojen pääperiaatteita. Järjestelmää alettiin kehittää ilman pilottiasiakasta. Tarkoitus oli rakentaa oma versio omilla ideoilla, ja vasta sitten esitellä tuote mahdolliselle pilottiasiakkaalle. Tässä vaiheessa itse yökerho toisi esille omat toiveensa järjestelmään liittyen ja kehitystyö jatkuisi yhteistyössä yökerhon kanssa. Tämän työn sisältö keskittyy nimenomaan kehitysvaiheeseen ennen pilottiasiakkaan hankkimista.

Järjestelmän kehittäminen oli myös oiva mahdollisuus tutustua julkisen web-sovelluksen kehittämiseen ja sen julkaisemiseen sekä ulkopuolisen webhosting-palvelun käyttämiseen palvelun teknisenä ylläpitäjänä. Webhosting on usein ainoa järkevä ja luotettava palvelinratkaisu nimenomaan yrittäjille ja pienille alan yrityksille, joilla ei ole varaa omaan palvelimeen ja sen luotettavaan ja turvalliseen ylläpitoon. Vaikka palvelut ovat usein tehty hyvin helppokäyttöisiksi asiakkalle, ei webhosting-palvelun ja webhotellin käyttöönotto ja käyttäminen ole täysin mutkatonta. (Heinisuo 2003, 19)

## 4.2Järjestelmän toimintaperiaatteet ja tavoitteet

Yksinkertaistettuna kehitettävän järjestelmän tehtävä on kerätä mahdollisimman paljon tietoa yökerhon asiakkaista ja heidän asiakaskäyttäytymisestään. Henkilötietojen tallentamisen ja hallinnan rinnalle halutaan mahdollisuus seurata asiakkaiden vierailu-

tiraidallista muovikorttia. Ostokäyttäytymistä ei järjestelmällä ainakaan aluksi kyetä seuraamaan, koska tekijöiden resurssit ovat rajalliset, eikä tarvittavaa tietoa ja taitoa järjestelmän liittämiseen kassajärjestelmään ole olemassa. Järjestelmän on toimittava myös markkinointikanavana kanta-asiakkaille. Lisäksi on kehitettävä sellaisia ominaisuuksia, joista myös yökerhon asiakkaat hyötyvät. Kehitettävä järjestelmä keskittyy siis asiakkuuden hallintaan ja seurantaan.

Asiakkaiden käyntiaikojen ja henkilötietojen perusteella kyetään kehittämään ja tehostamaan markkinointia ja myyntiä; saadaan jaettua eri aikoina vierailevia asiakkaita ikä- ja sukupuoliryhmiin ja vaikuttamaan näin oikeisiin asiakkaisiin oikealla tavalla. Järjestelmällä pystytään olemaan yhteydessä asiakkaisiin erillisellä web-sivustolla sekä sähköpostin välityksellä. Asiakaspalaute ja asiakkaiden mielipiteet voidaan hoitaa internetin välityksellä. Järjestelmän tehtävä on toimia työkaluna luotaessa tiiviimpää asiakassuhdetta yökerhon kanta-asiakkaisiin. Lisäksi tavoitteena on, että tapahtumien järjestäminen, niiden ajoittaminen ja kohdistaminen oikeille asiakasryhmille helpottuu, kun järjestelmästä aletaan saada haluttua tietoa.

Huomioon otettavaa on myös se, että koska järjestelmä tallentaa asiakkaan tiedot ja kuvan, vältytään kanta-asiakaskorttien väärinkäytöltä. Tavallisia, pahvisia tai muuten vaan ’tiedottomia’ VIP-kortteja, käyttävät yökerhot eivät ole kyenneet helposti tarkistamaan, onko käyttäjä kortin oikea omistaja, koska pahvikortilla ei usein näy edes kortinomistajan nimeä. Elektroninen järjestelmä tekee asiakkaan tunnistamisen helpoksi. Kun kortin näyttää järjestelmälle, asiakkaan kuva ja henkilötiedot ilmestyvät ruudulle.

Kun yökerho tai ravintola hyväksyy asiakkaan kanta-asiakkaakseen, saadaan haluttu hyöty vasta, kun asiakas alkaa käyttää kanta-asiakkuuden suomia etuja ja lisää käyntimääriään. Tämän takia on tärkeää, että kanta-asiakkuudesta on myös hyötyä asiakkaalle. Yökerhoalalla yleisin ’porkkana’ on kanta-asiakkaiden pääsy jonon ohi sekä halvemmat sisäänpääsymaksut ja joskus myös halvemmat juomat. Tämä asia ei ollut järjestelmän kehittäjien päätettävissä, vaan yökerho itse määrää asiakkaansa edut. Halusimme kuitenkin kehittää jotain myös asiakkaille. Ideoimme ja kehitimme toiminnon, jossa asiakas voi halutessaan näyttää ystävälleen läsnäolonsa yökerhossa. Tätä toimintoa päätettiin kutsua nimellä kaveritutka.

## 4.3Toteutustekniikat: mahdollisuudet ja kompastuskivet

Määriteltäessä järjestelmän käyttötarkoitusta yhdistettynä sen käyttöympäristöön oli alusta alkaen selvää, että järjestelmän toiminnoista suurin osa tulisi toimimaan web-ympäristössä. Asiakkaiden tulisi päästä helposti käsiksi omiin toimintoihinsa, ja yökerhon henkilökunnan sekä johdon kannalta olisi hyvä päästä järjestelmän toimintoihin käsiksi missä vain, milloin vain. Loppujen lopuksi päädyttiin yksinkertaisimpaan ratkaisuun, jossa koko järjestelmä toimisi web-palveluna. Web-palvelu on nykyään yleinen toteutustapa useissa erilaisissa tietojärjestelmissä nimenomaan käytettävyyden kannalta. Internet-yhteys riittää järjestelmän käyttämiseen. Tietoturvaongelmat ovat web-palveluiden merkittävin haittapuoli, mutta oikein toteutettuna järjestelmästä voidaan tehdä hyvinkin turvallinen.

### 4.3.1Ohjelmointikielen valinta

Vaihtoehtoja web-palvelun toteutustyökaluiksi oli muutamia, suurimpana rajoitteena luonnollisesti tekijöiden tietotaito. Web-palvelun ohjelmointikieleksi, jolla käyttöliittymä ja tiedon hallinta toteutettaisiin, oli kolme varteenotettavaa vaihtoehtoa: kaupallinen Microsoftin DOT NET -kehitysympäristö (.NET) sekä ei-kaupalliset Java ja PHP (Hypertext Preprocessor). Seuraavassa käydään läpi kaikkien kielten olennaisimmat ominaisuudet.

**DOT NET**

.NET Framework on Microsoftin kehittämä ohjelmistokomponenttikirjasto. .Net tukee useaa ohjelmointikieltä, joista käytetyimpiä ovat C# ja VB.Net. .Net mahdollistaa järjestelmien helpon, tehokkaan ja tietoturvallisen kehittämisen kohtuullisella ohjelmakoodi määrällä. Framework luokkakirjastot sisältävät muun muassa windows-, web-, web service- ja windows CE-ohjelmistojen kehittämiseen tarvittavia kirjastoja. Kielen suurin heikkous on työkalujen hinta. (Wikipedia 2008a)

**Java**

Java on Sun Microsystemsin kehittämä ohjelmointikieli, jota käytetään laajalti web-sovelluksissa. Kielen vahvuuksia ovat pitkälle kehittynyt tietoturva, sekä laaja muuttujien tyypitys, joka helpottaa virheenkorjausta. Kielen suurin heikkous on sen raskaus; Java vaati paljon ohjelmakoodia ja enemmän suoritustehoa laitteistolta kuin yksinkertaisemmat kielet, kuten PHP. (Wikipedia 2008b)

**PHP**

PHP on dynaamisissa web-sovelluksissa paljon käytetty komentosarjakieli. Tämä tarkoittaa, että kirjoitettu ohjelma käännetään ymmärrettävään muotoon vasta web-sivua kutsuttaessa. PHP:n etuja ovat sen kevyt rakenne ja suora käytettävyys MySql-tietokantojen kanssa. Heikkouksina voidaan pitää heikkoa muuttujien tyypittämistä, joka vaikeuttaa virheenkorjausta, sekä puutteita tietoturvassa. Tietoturvan taso on kuitenkin riittävä tietojärjestelmiin, jotka eivät sisällä erityisen salattavaa tietoa. (Heinisuo 2003, 16-17)

Koska budjetti järjestelmää varten oli käytännössä nolla, valinta supistui Javan ja PHP:n välille. Näistä kahdesta valittiin PHP. Suurimmat syyt valintaan olivat nimenomaan PHP:n ohjelmakoodin kevyt rakenne, hyvä yhteensopivuus tietokantaan sekä testiympäristön helppo rakentaminen. Käytettävissä olevaan aikaan nähden PHP oli myös kaukaa viisas ratkaisu. Java-järjestelmän ohjelmointi olisi vienyt enemmän aikaa. Tulimme siihen tulokseen, että tietoturva on riittävä verraten kehitettävän tietojärjestelmän sisältämän tiedon arvoon. Järjestelmä tulisi sisältämään asiakkaiden henkilötiedot sekä tiedot heidän käyntiensä ajankohdasta. Tällainen tieto ei ole erityisen herkkää eikä tietovuodon sattuessa aiheuta järjestelmän käyttäjille erityistä taloudellista vahinkoa.

### 4.3.2PHP:n historia ja ominaisuudet

PHP:n ensimmäinen versio, PHP/FI, oli Rasmus Lerdorfin 90-luvun puolessa välissä kehittämä yksinkertainen kieli lähinnä hänen omia tarkoitusperiään varten. Sen tärkeimmät toiminallisuudet olivat lomaketiedon käsittely sekä tuki mySQL-tietokantaa varten. Jo vuonna 1997 esiteltiin PHP:n versio 3.0, jonka kehittivät Andi Gutmans ja Zeev Suraski. Versio 3.0 sisälsi tuen useille eri tietokannoille, mm. MySQL:lle ja Oracle:lle. PHP 3.0 kehittyi ohjelmointiharrastajien suosikiksi sen avoimuuden ja laajennettavuuden ansiosta. Ohjelmistokehittäjät loivat omia laajennuksiaan ja näin kieli kehittyi jatkuvasti. Vuonna 2003 julkaistu PHP 4.0 käytti uutta ohjelmamoottoria, joka tarjosi parempaa suorituskykyä, luotettavuutta ja skaalautuvuutta, Apachen lisäksi tuen myös muille web-palvelimille sekä sisäänrakennetun sessionhallinnan ja paremman tuen olio-ohjelmointiin. Uusin versio 5.0 tarjoaa lisäksi täysin uudistetun olio-ohjelmointirakenteen, abstraktit luokat, paremman poikkeusten käsittelyn, yhtenäisemmän XML-tuen, edistyneemmän MySQL-tuen sekä paremman muistinhallin-

nan. Netcraftin vuonna 2004 tekemän tutkimuksen mukaan yli 15 miljoonaa web-sivustoa käytti PHP-kieltä. (Wasvani 2005, 8-9)

**Ominaisuudet ja vahvuudet ohjelmointikielenä**

PHP on palvelinpuolen web-ohjelmointikieli. Tämä tarkoittaa, että asiakaskoneen pyytäessä web-dokumenttia verkon yli palvelimelta, palvelin käsittelee dokumentissa olevan PHP-koodin ennen dokumentin lähettämistä asiakaskoneelle. PHP-koodi on upotettu tavallisen HTML-koodin sekaan. Dokumentti lähetetään asiakaskoneelle tavallisena HTML-kieleisenä dokumenttina, PHP-koodia on käytetty vain sen sisällön muokkaamiseen annettujen ehtojen mukaisesti. PHP:lla voidaan luoda tavallisiin web-dokumentteihin dynaamista sisältöä ja sillä voidaan käyttää suurinta osaa ohjelmointikielien perusfunktioista, kuten ehtolauseita ja silmukkarakenteita. Lisäksi PHP:n vahvuuksiin voidaan laskea sen yhteensopivuus ja käytettävyys MySQL-tietokantojen kanssa. Näiden kahden tekniikan yhdistelmä onkin yksi käytetyimmistä ilmaisista työkaluista tietokantapohjaisissa web-sovelluksissa. (Vaswani 2005, 9)

**Yksinkertaisuus**

PHP käyttää johdonmukaista, loogista merkintätapaa. Selkeä manuaali ja tutoriaalit tekevät siitä helposti opittavan kielen myös aloittelijoille. PHP voi käsitellä esimerkiksi C-kielen kirjastoja ja käyttää hyväksi tällä kielellä kirjoitettua ohjelmakoodia. (Vaswani 2005, 9-10)

**Siirrettävyys**

PHP:n käyttäjien ei tarvitse juurikaan huolehtia käytettävästä ohjelmistoalustasta. PHP toimii UNIX:n, Microsoft Windowsin, Mac OS:n ja OS/2:n päällä. Koska PHP-koodia ei tarvitse kääntää, toimivat kirjoitetut skriptit lähes poikkeuksetta samanlaisina siirryttäessä alustalta toiseen. Käytännössä tämä näkyy myös meidän työskentelyssämme. Järjestelmä kehitetään Windows-ympäristössä ja valmis ohjelma siirretään UNIX/Linux-ympäristöön. Ohjelma toimii silti samoin. (Vaswani 2005, 9-10)

**Nopeus**

Jo käyttöönotossa PHP toimii nopeammin kuin useimmat skriptipohjaiset ohjelmointikielet, kuten JSP (Java Server Pages), ASP:NET ja Perl. Jo versio 4.0 nosti suoritustasoa tällä rintamalla, ja 5.0 versio paransi suorituskykyä entisestään optimoidun muistinhallinan ja muistia säästävän oliokäsittelyn ansiosta. (Vaswani 2005, 9-10)

**Avoin lähdekoodi**

PHP on ilmainen käyttää ja sen lähdekoodi on vapaasti saatavilla internetistä. Näin ollen PHP voi tuntuvasti vähentää järjestelmäkehityksen ja järjestelmäylläpidon kustannuksia. Avoin lähdekoodi mahdollistaa myös nopeamman virheiden korjaamisen ja integroinnin uusiin tekniikoihin, koska käyttäjä- ja kehittäjäkanta on valtavan suuri. (Vaswani 2005, 9-11)

**Laajennettavuus**

PHP:n kehittäjät pitivät tulevaisuuden laajennukset mielessään rakentaessaan kielen arkkitehtuuria. Ohjelmistokehittäjät ympäri maailman voivat luoda kieleen laajennuksia ja lisäosia uusia tekniikoita varten moduuleina. Esimerkkinä luoduista moduuleista mainittakoon dynaaminen kuvien, PDF- ja SWF-tiedostojen luominen, yhteys IMAP ja POP3 -sähköpostipalvelimiin, rajapinta MySQL, Oracle, PostgreSQL ja SQLite -tietokantoihin sekä XML-dokumenttien parsinta. PHP:n erillisistä lisäosista on kasattu myös kattava ja ilmainen paketti, PEAR (PHP Extension and Application Repository), joka sisältää uudelleen käytettäviä, virhekorjattuja PHP-komponentteja. (Vaswani 2005, 10-11)

**XML- ja tietokantatuki**

Käyttipä kehitettävä järjestelmä tiedonhallintaan sitten XML-kieltä tai tietokantaa, PHP käsittelee molempia. PHP-versio 5.0 tukee MySQL:n lisäksi DB2, PostgreSQL, Oracle, mSQL, MS-SQL, Informix, Sybase sekä SQLite -tietokantoja. Siitä löytyy XML-ohjelmointirajapinta, jonka mukana tulevat SAX, DOM, XSLT, SimpleXML sekä SOAP -laajennukset. (Vaswani 2005, 11)

### 4.3.3Tiedon varastointitekniikka

Tiedon varastointiin oli käytännössä kaksi vaihtoehtoa: tietokanta, joka on ollut tiedonvarastoinnin standardi jo vuosia, ja XML (Extensible Markup Language), joka on tulevaisuuden kieli tiedon käsittelyssä. Pelkän XML-tiedon käyttäminen ei kuitenkaan tuntunut järkevämmältä vaan pikemminkin hankalammalta vaihtoehdolta, koska työn tekijät eivät olleet tutustuneet kieleen kovinkaan hyvin ja monet sen ominaisuudet olivat vielä tuntemattomia. Myös XML-kielen tietoturva-asiat olivat epäselviä. Aluksi ajateltiin käyttää molempia vaihtoehtoja, mutta helpottaaksemme työtämme ja keskittääksemme tiedon, päädyttiin käyttämään pelkkää tietokantaa. Myös molempien tekijöiden käytännön kokemus tietokantojen käytöstä puolsi valintaa. Tietokantoja on

useaa eri tyyppiä. Yleisimpinä mainittakoon Microsoft Access, Oracle ja MySql. Kaikki mainitut käyttävät SQL (Structured Query Language) -tietokantakieltä, joka on hyvin yleinen standardi. Näistä kolmesta MySql on ainut käytössä ilmainen tietokanta, joka on lisäksi hyvin yhteensopiva PHP-kielen kanssa. MySql on maailman yleisin vapaan lähdekoodin omaava tietokantamoottori, joten luotettavuuskin oli kohdallaan. Päätettiin käyttää MySql -tietokantaa.

### 4.3.4MySQL:n historia ja ominaisuudet

MySQL on erittäin suorituskykyinen relatiivinen tietokantahallintajärjestelmä, jota voidaan pitää tietokantapohjaisten sovellusten standardina sekä web-pohjaisissa kuin muissakin ohjelmissa. Sen suunnittelun pääperiaatteet ovat nopeus, vakaus ja helppokäyttöisyys. Viralliset tilastot kertovat yli viiden miljoonan sivuston käyttävän mySQL-pohjaisia sovelluksia. (Vaswani 2005, 11)

MySQL:n juuret sijoittuvat vuoteen 1979, jolloin Michael Widenius loi UNIREG-nimisen tietokantajärjestelmän ruotsalaiselle TcX-yhtiölle. UNIREG ei toiminut toivotulla tavalla, joten Widenius alkoi kehittää uutta järjestelmää mSQL-rajapintaa hyväksi käyttäen. Tuote valmistui vuonna 1996 nimellä MySQL 1.0. Vain muutamia kuukausia myöhemmin MySQL 3.11 julkaistiin binäärijakeluna Solarikselle. Lähdekoodi ja binäärijakelu Linuxille julkaistiin pian tämän jälkeen. TcX:stä tuli MySQL Ab, yksityinen yritys, joka omistaa kokonaan MySQL:n palvelinlähdekoodin ja tuotemerkin. Tänä päivänä MySQL on saatavissa useille eri alustoille mukaan lukien Linux, Mac OS ja Windows. (Vaswani 2005, 11-12)

**Nopeus**

Relatiivisten tietokantojen tärkein ominaisuus on niiden nopeus; se, kuinka paljon aikaa kuluu tietokantakyselyn suorittamiseen ja tulosten palauttamiseen. Tällä rintamalla MySQL on edellä suurinta osaa kilpalijoitaan, kuten Microsoft SQL Serveriä ja IBM DB2:ta. (Vaswani 2005, 12-13)

**Luotettavuus**

MySQL-relaatiotietokantaa on testattu korkeilla käyttäjämäärillä kriittisissä tehtävissä maailman suurimmissa yrityksissä, kuten NASA:ssa, HP:lla ja Yahoo:lla. Koska lähdekoodi on avoin, uusia versioita ovat testanneet käyttäjät ympäri maailman erilaisissa

olosuhteissa ja tilanteissa, jotta on voitu varmistaa kunkin uuden version virheettömyys ja luotettavuus. (Vaswani 2005, 12-13)

**Tietoturva**

MySQL:n kehittäjät ovat varmistaneet tietokannan tietoturvan kehittyneellä käyttöoikeushallinnalla väärinkäytösten ehkäisemiseksi. MySQL-ylläpitäjät voivat suojata arkaluonteisen tiedon käyttäen käyttäjän ja palvelimen välillä toimivaa autentikointia. Tietyn käyttäjän oikeuksia voidaan rajoittaa esimerkiksi vain tiettyihin tietokantoihin tai tauluihin tai jopa tietyn tyyppisiin hakuihin tietyistä tauluista. (Vaswani 2005, 13-14)

**Vakaus ja siirrettävyys**

MySQL kykenee käsittelemään erittäin suuria ja monimutkaisia tietokantoja ilman suurta vaikutusta suorituskykyyn. Useamman gigatavun kokoiset taulut, jotka sisältävät satojatuhansia rivejä, eivät ole harvinaisia. Kun tauluihin on saatu tietoa, voi tietokannan siirtää alustalta toiselle vaikeuksitta. MySQL on saatavilla niin UNIX-pohjaisille kuin muillekin alustoille mukaan lukien Linux, Solaris, FreeBSD, OS/2, MacOS sekä Windowsin versiot 95, 98, Me, 2000, XP ja NT. MySQL toimii myös hitaammilla tietokoneilla aina 386-prosessoreista lähtien. (Vaswani 2005, 13-14)

**Yhteensopivuus olemassa olevien standardien kanssa**

MySQL 4.0 tukee tärkeimpiä ANSI SQL-99 -tietokantastandardin ominaisuuksia. MySQL myös laajentaa ANSI-standardia omilla funktioilla ja tietotyypeillä, jotka parantavat nimenomaan siirrettävyyttä ja antavat enemmän toimintamahdollisuuksia käyttäjälle. Omassa järjestelmässämme yhteensopivuus saattaa olla hyödyksi tulevaisuudessa, mikäli järjestelmä siirretään toiseen käyttöympäristöön ja eri tietokantamoottorin alle. Näin ollen samainen tietokanta tulee toimimaan ilman suurempia muutoksia missä tahansa muussa ANSI-standardin mukaisessa tietokantaympäristössä. (Vaswani 2005, 14-15)

**Laaja sovellustuki**

MySQL sisältää yhteysrajapinnan useimpiin ohjelmointikieliin. Näin ollen MySQL-tietokantaa käyttäviä soveluuksia ja järjestelmiä voidaan kirjoittaa useilla eri kielillä ja itse tietokanta on käytettävissä samanlaisena ohjelmointikielestä riippumatta. PHP-tuen lisäksi MySQL Ab on julkaissut ODBC ja JDBC -ajurit Microsoft Windows ja Java -alustoille. Rajapinnat löytyvät myös C, C++, Perl, Python sekä Tcl -kielille.

Omaa järjestelmäämme ajatellen on hyvä asia pitää mielessä, että tietokantaa ei välttämättä tarvitse rakentaa uusiksi, vaikka järjestelmä toteutettaisiinkin tulevaisuudessa raskaammalla ohjelmointikielellä. (Vaswani 2005, 14-15)

### 4.3.5Web-sivustojen käytettävyys

Järjestelmän käyttäjäkunta tulee olemaan laaja. Sen on tarkoitus palvella niin ravintolan kuin ravintolan asiakkaidenkin etuja. Käyttäjäkunta on laaja, koska käytännössä kenellä tahansa on mahdollisuus päästä järjestelmän asiakkaaksi. Lisäksi ravintolahenkilökunta on myös vaihtuva käyttäjäkunta Työntekijät vaihtuvat ja tuskin minkään ravintolan henkilökunta pysyy samana avajaisista sulkemiseen. Vaikka kyseessä onkin asiakashallintajärjestelmä, on järjestelmä samalla myös web-sivusto. Käytettävyyden tulee olla tarkoin mietitty ja siinä täytyy ottaa huomioon niin yleiset web-sivustojen käytettävyyteen liittyvät asiat kuin tietojärjestelmän rakentamisessa huomioon otettavat logiikatkin. John Cato (2001) kirjoittaa teoksessaan User-Centered Web Design, että käytettävyysasioiden on oltava mielessä koko sivuston kehitysprojektin ajan. Käytettävyyden tulisi perustaa tietyille metodeille, joita noudatetaan koko työskentelyn ajan. Cato loi viidestä askeleesta koostuvan lähestymistavan, jota tässä projektissa suurelta osin käytettiin. (Cato 2001, 69-73)

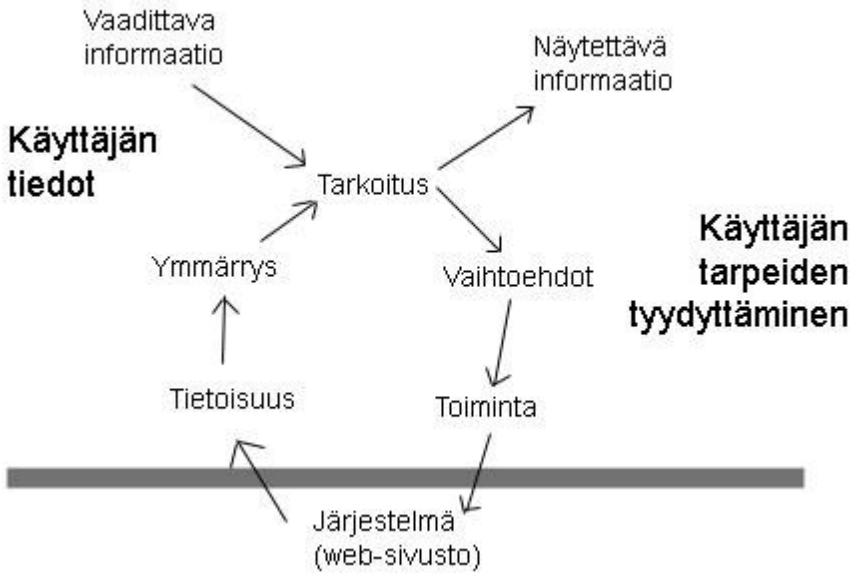
Ensimmäinen askel on alueet, joihin web-sivu on jaettu. Perusidea on jakaa sivu osioihin, jotka sisältävät toimintaa, sekä osioihin, jotka sisältävät informaatiota. (Cato 2001, 73-74)

Toinen askel on tiedostaa sivuston koko sisältö ja rakenne eli se, mitä eri sivuja tarvitaan ja miten eri sivujen välillä liikutaan. Cato suosittelee kirjoittamaan niin sanotun storyboardin sivujen välillä liikkumisesta ja eri toiminnoista. Tietojärjestelmäprojektissa saman asian ajavat käyttötapauskaaviot. Tärkeimmät kulmakivet ovat koko sivuston rakenne, kussakin tilanteessa käytettävissä olevat navigaatiot ja toiminnalliset kontrollit sekä näytettävä informaatio. (Cato 2001, 73, 77-80)

Kolmas askel on miettiä, mikä on kunkin sivun sisältö ja mitä sivun eri alueet eri tilanteissa sisältävät. Myös tässä suhteessa tietojärjestelmä eroo tavanomaisesta informatiivisesta web-sivusta. Järjestelmässämme on useissa käyttötilanteissa dynaamista informaatiota ja sivun eri alueiden sisältämä tieto ja toiminnallisuudet vaihtelevat tilanteen mukaan. (Cato 2001, 73, 83)

Neljäs askel on vuorovaikutuksen säilyttäminen käyttäjän ja järjestelmän välillä. Sivulla näkyvän tiedon ja toimintavaihtoehtojen tulee antaa käyttäjälle tarvittava tieto ja mahdollisuudet toimintojen suorittamiseen ja informaation saamiseen. Järjestelmä johdattelee käyttäjän tekemään haluttuja toimintoja eikä sekoita käyttäjän ajatuksia näyttämällä harhaanjohtavaa informaatiota tai toimintavaihtoehtoja. Vuorovaikutuksen kehittämiseen Cato esittelee kirjassaan AUA-mallin (Awareness, Understanding, Action). (Cato 2001, 73, 93-97)

KUVIO 2. John Caton AUA-malli (Cato 2001, 95)




AUA-malli esittää yksinkertaisuudessaan sen, miten web-sivuston informaatio on vuorovaikutuksessa käyttäjän ja hänen tietojensa kanssa. Käyttäjän tiedot ohjaavat sivuston toimintaa ja taas päinvastoin sivuston toiminta ohjaa käyttäjän toimintaa ja hänen tietoisuuttaan. Huonosti rakennettu toiminto sivustolla voi johtaa siihen, että käyttäjä ja järjestelmä eivät ole enää vuorovaikutuksessa toistensa kanssa, mistä lopputuloksena on väärä toiminta tai puutteellinen tai kokonaan väärä näytettävä informaatio. (Cato 2001, 95-97)

Viimeinen, viides askel, on järjestelmän visuaalinen tyyli. Sivuston tulee näyttää ja miellyttää käyttäjää juuri sillä alalla, johon se on suunnattu. (Cato 2001, 73)

Näiden perusasioiden mukaan toteutettiin järjestelmän rakenne ja toiminnallisuudet. Jokainen toiminto pyrittiin ajattelemaan käyttäjän näkökulmasta niin, että käyttäjän mielenkiinto pysyisi toiminnossa, ja käyttäjälle annettaisiin tarvittava informaatio kunkin toiminnon käyttämiseen. Erilaisia käyttötilanteita ja toimintavaihtoehtoja niissä on kuvattu käyttötapauksina (liite 3) ja käyttötapauskaaviona (liite 2).

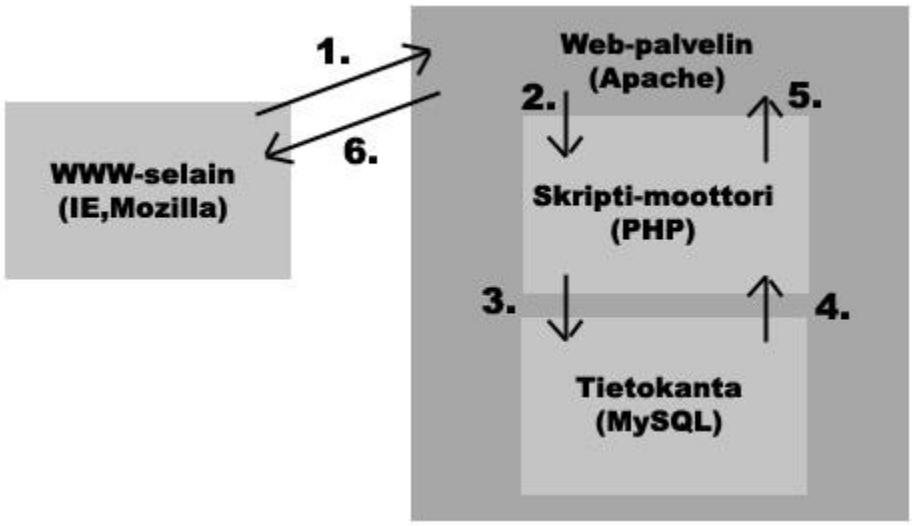
# 5 JÄRJESTELMÄN TOTEUTUS

Kun tekniikat ja työkalut järjestelmän toteuttamiseksi on valittu, on tärkeää miettiä, miten suunnittelu- ja ohjelmointivaiheet tulevat etenemään. Tekemällä asioita oikein alusta lähtien säästetään usein ajan lisäksi myös rahaa. Tämä projekti on kuitenkin tehty hyvin pienellä budjetilla, joten taloudellisia kysymyksiä mietittiin lähinnä periaatetasolla. Tärkeimmät askeleet siirrettäessä ideat julkiseksi tietojärjestelmäksi ovat järjestelmän huolellinen suunnittelu, toimivan järjestelmän toteuttaminen ja koodaaminen sekä julkisen järjestelmän rakentaminen ja fyysinen sijainti, tässä työssä webhotelli.

## 5.1Tietokantapohjaisten verkkosovellusten arkkitehtuuri

Web-sovelluksen toimintaperiaate on yksinkertaistettuna seuraava: asiakas (engl. Client) eli sovelluksen käyttäjä pyytää web-palvelimelta haluamaansa dokumenttia, jonka web-palvelin lähettää asiakkaalle. Asiakas on yleensä käyttäjän käyttämä internet-selain ja dokumentti HTML-muodossa oleva tiedosto. Saatuaan pyytämänsä dokumentin web-palvelimelta selain näyttää dokumentin sisällön. Tapahtuma voi olla näin yksinkertainen, mikäli halutut dokumentit ovat sisällöltään staattisia eivätkä sisällä dynaamista tai tilanteen mukaista muokkausta. Tämä pyyntö-vastaus-toimenpide ei kuitenkaan ole palvelimen puolella aina näin yksinkertainen. Useimmat kehittyneemmät web-palvelimet ja -sovellukset käyttävät jotakin ohjelmointi- tai skriptikieltä muokkaamaan pyydetyn dokumentin sisällön tiettyyn muotoon, eli dokumentti sisältää dynaamista sisältöä. Lisäksi käytössä voi olla tietokanta tiedon varastointia varten.

KUVIO 3. Verkkosovelluksen arkkitehtuuri. (Heinisuo 2003, 12)




Dynaamista tietoa sisältävä dokumentti käy läpi kuvion 3 mukaiset vaiheet. Vaiheita voi olla enemmän tai vähemmän riippuen käytettävästä palvelimesta, ohjelmointikielestä (skriptikielestä) ja tietokannasta. Tämä on kuitenkin yleisin toimintaperiaate web-sovelluksissa ja tähän arkkitehtuuriin perustuu myös tämän työn tietojärjestelmä. Vaiheessa 1 selain pyytää tarvittavaa dokumenttia verkon yli web-palvelimelta. Palvelin huomaa pyydetyn dokumentin sisältävän skriptattua sisältöä ja antaa dokumentin käännettäväksi skriptimoottorille (vaihe 2, esimerkissä PHP). Kääntäessään skriptiä PHP-moottori huomaa dokumentin tarvitsevan tietoa tietokannasta tai dokumentti haluaa muokata tietokannan tietoja. Näin ollen PHP-moottori pyytää tietokantamoottoria käsittelemään tarvittavat tiedot (vaihe 3, esimerkissä MySQL-tietokanta). MySQL suorittaa pyydetyt toimenpiteet ja palauttaa tiedon PHP-moottorin käsiteltäviksi (vaihe 4). PHP-moottori käsittelee saamansa tiedon ja muodostaa HTML-dokumentin, jonka palauttaa web-palvelinohjelmalle (vaihe 5). Nyt asiakkaan pyytämä dokumentti on käynyt läpi tarvittavat muutokset, ja web-palvelin voi lähettää muodostetun HTML-dokumentin takaisin selaimelle (vaihe 6). Lopuksi selain näyttää HTML-dokumentin käyttäjälle. (Heinisuo 2003, 11-15; Welling & Thomson 2001, 180-181.)

## 5.2Tietokantapohjaisen web-sovelluksen kehitys- ja käyttöympäristö

Kehitettäessä web-sovellusta ohjelmaa ei voida suoraan ohjelmoida ohjelmointikieltä ja tietokantatyökaluja käyttäen ja viedä sitä lopulliseen käyttöympäristöön, vaan suurin osa ohjelman kehityksestä tapahtuu kehitysympäristössä. Web-sovelluksista puhuttaessa tarkoitetaan palvelinalustaa, joka on vain projektiryhmän käytössä, ja jossa ovat

samat palvelinohjelmat ja työkalut kuin varsinaisessa käyttöympäristössäkin. Lisäksi kehitystä varten on olemassa salasanasuojattu web-hotelli, jossa on lähes sama kokoonpano kuin työasemillakin. Itse käyttöympäristö tulee olemaan web-hotellissa, omalla yksityisellä palvelimella tai jaetulla palvelimella asiakkaan toiveiden mukaisesti.

### 5.2.1Apache web-palvelinohjelmisto

Web-sovellus tarvitsee toimiakseen palvelinohjelman, joka lähettää halutun dokumentin asiakkaan koneelle asiakkaan pyynnöstä. Palvelinohjelma voi toimia yksinkertaisesti vain HTML-dokumenttien lähettäjänä, mutta se osaa tarvittaessa käsitellä myös lisäosina asennettuja web-ohjelmointikieliä. Koska ohjelmointikieleksi oli valittu PHP, oli myös palvelinohjelman vaatimuksena PHP-tuki. (Heinisuo 2003, 14-15)

Suosituimpia PHP-tuellisia palvelinohjelmistoja ovat Microsoft IIS (Internet Information Server) ja Apache. Molemmat ovat laajalti käytettyjä ja luotettavia ohjelmistoja. Microsftin IIS on luonnollisesti suunniteltu Windows-ympäristöön, kun taas Apachen etuna on sen toimivuus sekä Unix -järjestelmissä että Windows -ympäristössä. Tässä projektissa vielä suuremmaksi Apachen eduksi nousi hinta. Apache on ilmainen, IIS ei ole. (Heinisuo 2003, 20-21)

Kun tähän lisätään vielä se etu, että suurimmat Webhosting-palvelut tarjoavat palvelunsa Unix -alustalle asennetulla Apachella, ei palvelinohjelmiston valinnassa ollut epäselvyyksiä. Se, että kehitysympäristössä omilla työasemilla käytettiin Apachea Windowsin kanssa ja Webhosting-palvelut toimivat Unix -pohjalla, ei vaikuttanut millään tavalla siirryttäessä kehitysympäristöstä lopulliseen käyttöympäristöön. (Greenspan & Bulger 2001.)

### 5.2.2Kehitysympäristö

Kehitysympäristönä käytettiin omille työasemille asennettua Apache-palvelinohjelmaa PHP- ja MySQL-tuella varustettuna. Apache toimi työasemilla vain paikallisena, eli se ei lähettänyt tietoa tietokoneen ulkopuolelle, vaan vain paikallisen internet-selaimen pyynnöstä. Tällä tavalla tehtiin ohjelmointityötä itsenäisesti ohjelmoiden vain sovittuja osia omalla tietokoneella.

Lisäksi järjestettiin testiympäristö web-hotelliin, johon pääsi käsiksi myös verkon yli. Kehitysvaiheessa tämä sivusto oli suojattu salasanalla mutta vastasi muuten tulevaa käyttöympäristöä. Suurimpana erona työasemalla paikallisena toimivan testiympäristön ja Webhosting-palvelun julkisen ympäristön välillä oli palvelinohjelmiston asetusten muokattavuus. Omilla työasemilla saatettiin muokata kaikkia mahdollisia Apachen asetuksia, mutta Webhosting-palvelun asetukset olivat vakiot, eikä niitä voitu muuttaa. Olennaisimpia eroja olivat virheilmoitukset ja PHP:n asetuksiin liittyvä globaalien muuttujien asetus.

Julkinen web-hotelli ei näytä virheilmoituksia käyttäjälle virheen sattuessa, kun taas omalle työasemalle saatettiin laittaa tämä asetus päälle. Kehitysvaiheessa tämä on tärkeimpiä ominaisuuksia, joita ohjelmoija tarvitsee testatessaan ohjelmakoodin toimivuutta. Kun palvelinohjelma palauttaa virheilmoituksen, tiedetään melko tarkkaan, missä mahdollinen vika sijaitsee.

Globaalien muuttujien yhteydessä tilanne on mutkikas. Globaali muuttuja on PHP-kielen tai muun web-ohjelmointikielen muuttuja, joka on esimerkiksi lomakkeelta syötetty POST-muuttuja, url-osoitteesta saatava GET-muuttuja tai käyttäjän istunnon ajan voimassa oleva SESSION-muuttuja. Yleinen suositus tietoturvan takia on, että globaalit muuttujat olisivat poissa käytöstä. Tällöin muuttujat eivät toimi ohjelmakoodissa tavallisten muuttujien tavoin, vaan ne pitää aina erikeen noukkia POST-,GET- tai SESSION-metodeista. Näin väärinkäyttötarkoituksessa syötetyt muuttujat lomakkeiden kautta tai url-osoitteen jatkeena eivät suoraan voi aiheuttaa vahinkoa ohjelmakoodin sisällä ja vain ohjelmakoodissa tarkoituksellisesti poimitut muuttujat käytetään suoritettavan tehtävän käsittelyssä. (PHP Security Guide 2005, kappale 1.3)

Webhosting-palvelut laittavat palvelimilleen globaalit muuttujat kuitenkin usein päälle, koska monet PHP-harrastelijat ja kokemattomammat ohjelmoijat ohjelmoivat sivustonsa ilman, että poimivat globaaleja muuttujia. Näin on nopeampaa ja helpompaa ohjelmoida, mutta ohjelmointi tapahtuu tietoturvan kustannuksella. Tämän vuoksi tuli kiinnittää erityistä huomiota muuttujien poimimiseen ja ennen kaikkea niiden nimeämiseen, jotta tietoturva-aukoilta vältyttäisiin. Ongelma aiheutti myös ristiriitatilanteita ohjelmakoodin sisällä, kun web-hotellin palvelin ymmärsi tavalliset muuttujat esimerkiksi SESSION-muuttujina, mikäli kyseinen SESSION-muuttuja oli olemassa. Tämä ongelma saatiin kuitekin selvitettyä ja ohitettua tarkoilla muuttujien nimien määrittelyillä. (Lehtonen, 2007)

### 5.2.3MySQL kehitysympäristössä ja web-hotellissa

Apache sisältää tuen MySQL-tietokannalle ja PHP:lla on suora MySQL-tuki, eli tietokannan käyttäminen ohjelmakoodissa oli vaivatonta. Omalla työasemalla riittää, että Apachen MySQL-tuki on otettu käyttöön ja PHP-asetuksissa on MySQL-tuki käytössä sekä MySQL on asennettu tietokoneelle. Tietokantaan pääsee käsiksi paikallisesti käyttäen mitä tahansa tietokantatyökalua, esimerkiksi MySQL-Frontia tai DBManageria.

Web-hotelleissa on myös usein MySQL-tuki ja myös tässä projektissa käytetystä webhotellista tämä tuki löytyi. Kun MySQL-palvelin on julkinen, on se myös alttiimpi väärinkäytölle kuin paikallinen, työasemalla toimiva MySQL-tietokanta. Edelleen on Webhosting-palveluja, jotka käyttävät täysin erillistä julkista MySQL-palvelinta, johon otetaan suojaamaton yhteys web-palvelimelta. Tämä tarkoittaa, että tietokantapalvelimen osoite, käyttäjätunnus ja salasana liikkuvat verkossa salaamattomina, ja ovat näin osaavien väärinkäyttäjien kaapattavissa. Luotettavimmat ja turvallisimmat Webhosting-palvelut ovat kuitenkin siirtyneet jo tarjoamaan tietokantapalvelua, joka toimii paikallisena web-palvelimella. Näin tietoa ei liiku verkossa, vaan palvelimen sisällä, jolloin tunnukset, salasanat ja tieto eivät pääse vääriin käsiin.

Vanha tapa oli siinä mielessä kätevä, että tietokantaan pääsi käsiksi vain antamalla tietokantatyökalulle tietokantapalvelimen osoitteen, käyttäjätunnuksen ja salasanan. Paikalliseksi web-palvelimelle asennettuun tietokantaan ei pääse käsiksi tällä tavalla, vaan tietokantayhteys täytyy niin sanotusti tunneloida. Tämä tapahtuu salatun yhteyden avulla, esimerkiksi ilmaisella Putty Link -ohjelmalla. Salattu yhteys vaatii usein web-hotellilta SSH-tuen. Web-palvelimelle muodostetaan yhteys tiettyyn porttiin, ja tämä portti yhdistettään oman tyäaseman haluttuun porttiin. Näin web-palvelimella oleva tieto näkyy valitun portin kautta omalla tietokoneella paikallisena, ja tietokantayhteyden luonti voidaan suorittaa paikallisena tähän porttiin. Tarkempia tietoja tästä toimintatavasta ja yhteyden luomisesta on osoitteessa http://www.louhi.net/faq?cat=1 (Louhi Net 2008).

### 5.2.4Lopullinen käyttöympäristö

Järjestelmä ohjelmoitiin osissa niin, että toiminto todettiin ensin toimivaksi omalla työasemalla, minkä jälkeen se liitettiin web-hotellissa olevaan testijärjestelmään. Täs-

sä vaiheessa testattiin toiminnon toimivuus myös web-hotellissa, jolloin toiminto oli käytännössä valmis käytettäväksi lopullisessa järjestelmässä. Web-hotellissa sijaitseva testiympäristö toimi yhtä aikaa projektin keskipisteenä sekä viimeisenä testiversiona lopullisesta järjestelmästä.

## 5.3Ohjelmointikäytännöt ja metodit järjestelmää toteutettaessa

Tietojärjestelmän kehitystyössä tärkeimmät päätökset koodirakenteen ja ohjelmointimetodien suhteen tulee tehdä heti ohjelmointivaiheen alussa. Tämä sen takia, että järjestelmän ohjelmointi on tavallaan kuin saman puun veistämistä. Kun ohjelmakoodia kirjoittaa useampi kuin yksi henkilö useammassa kuin yhdessä paikassa, on lopputulos kaksi puolikasta ohjelmaa, mikäli toimintatavoista ei sovita etukäteen. Tämä projekti käynnistyi juuri tähän, väärään suuntaan, mutta jo alkumetreillä todettiin, ettei työ etene toivotulla tavalla. Seuraavaksi käydään läpi tärkeimmät työssä ja työskentelyssä käytetyt tekniikat ja käytännöt.

PHP-kieleen on sisällytetty olio-ohjelmointimahdollisuus sen versiosta 5.0 alkaen. Kun järjestelmän toiminnot käyttävät suurelta osin hyväkseen tietokantaa, on toiminnot järkevintä toteuttaa luokkapohjaisella ohjelmarakenteella. Yleisin tapa on määritellä jokaiselle ohjelman sisäiselle toimivalle ja muuttuvalle yksiköllе oma luokkansa. Tässä projektissa päätettiin kuitenkin ottaa toinen, vielä yksinkertaisempi lähestymistapa. Luokiksi määritettiin tiedonkäsittelyyn liittyvät toiminnot: tiedon lisääminen, tiedon muokkaus, tiedon poistaminen ja tiedon hakeminen sekä yksi apuluokka, jolla oli vapaampi rooli tietoa käsiteltäessä. Tähän oli kaksi keskeistä syytä. Koska järjestelmä ei ole sisällöltään kovin laaja johtuen työryhmän koosta, valittu luokittelu on riittävä pitämään ohjelmoinnin systemaattisena ja jäsenneltynä. Lisäksi tiedossa oli yhteisten kontaktikertojen vähäisyys ja se, että toteutus tulisi tapahtumaan suureksi osaksi etätyöskentelynä. Näin ollen toimintopohjainen luokittelu antoi molemmille ohjelmoijille vapaammat kädet luoda ohjelman toiminnallisuutta verrattuna suurempaan luokkamäärään. Tietojen yhtenäisyyden korjaamista tarvitaan vähemmän valitulla metodilla. Kuitenkin ajatustasolla, ideointi- ja kehitysvaiheessa, pidettiin eri yksiköitä omina luokkinaan ja niiden sisältämää tietoa ja elinkaarta käsiteltiin aivan kuten olio-ohjelmoinnissa yleensä. (Vesterholm & Kyppö 2006, 79-81)

### 5.3.1Järjestelmän visuaalinen kehitys ja käyttöliittymän rakentuminen

Järjestelmää alettiin toteuttaa suunnittelemalla alustava käyttöliittymä ja valikkorakenne, jolla ohjelman työkaluja ja toimintoja olisi helppo ja nopea käyttää. Työn tekijöiden omien ideoiden ja aiemman ohjelmointi- ja suunnittelukokemuksen ansiosta päästiin hyvin nopeasti vaiheeseen, jossa valikkorakenne ja järjestelmän toimintalogiikka olivat selvillä. Tiettyjä näkemyseroja syntyi joidenkin toimintojen toteutuksen osalta, koska emme olleet aiemmin työskennelleet samoissa projekteissa. Nämä olivat kuitenkin hyvin vähäisiä käytettävyyteen liittyviä mielipiteitä, joista päästiin eteenpäin kompromisseilla puoleen ja toiseen.

Sivuston aluejako on yksinkertainen. Sivun ylälaidassa on web-sivustoille tyypillinen yläbanneri, joka pysyy paikallaan koko ajan. Bannerin kuva luo sivuston perusilmeen muttei kuitenkaan vie liikaa tilaa loppusivulta. Järjestelmä käyttää navigointiin sivun vasemmassa laidassa jatkuvasti pysyvää päävalikkoa. Erillistä sivuvalikkoa ei ole, vaan sivuvalikon työn ajaa päävalikon dynaamisuus, ja tietyissä toiminnoissa päävalikkoon tulee lisää valittavaa. Loppualue sivusta on tarkoitettu informaatiolle ja tietyille toiminnoille informaation yhteydessä.

Kokonaisuuden hahmottamista auttavat käyttötapauskaaviot (liite 2) tärkeimmistä toiminnoista. Toiminnot toteutettiin niin, että käyttäjä voi aina perua toiminnon tai palata taaksepäin ilman selaimen Back-painiketta. Kulloinkin näkyvissä olevia toimintamahdollisuuksia pyrittiin rajaamaan ja porrastamaan niin, ettei sivulla olisi kuitenkaan liikaa toimintamahdollisuuksia. Näin ollen käyttäjän on helppo lähteä suorittamaan haluamaansa tehtävää tai hakemaan haluamaansa tietoa. Toimintojen ja toimintamahdollisuuksien hahmottamisessa auttavat myös käyttötapauskuvaukset (liite 3).

### 5.3.2Järjestelmän teknisiä yksityiskohtia

Suuri osa järjestelmästä oli yksinkertainen toteuttaa. Projektiparilla oli aiempaa kokemusta tietokantapohjaisista web-järjestelmistä ja selkeä käsitys siitä, mitä oltiin kehittämässä. Jokainen järjestelmä on kuitenkin aina ainutlaatuinen silloin, kun sitä lähdetään tekemään alusta loppuun. Ohjelmoinnissa voidaan käyttää runsaasti olemassa olevia käytäntöjä ja valmiita komponentteja. Tässäkin järjestelmässä oli kuitenkin muuta-

mia teknisiä seikkoja, joissa oli useita erilaisia toteutusmahdollisuuksia. Näistä mainittakoon käyttäjäoikeudet (ravintolahenkilökuntaan kuuluvan käyttäjän käyttäjäoikeudet) sekä asiakaskuvien tallennustapa.

Henkilökuntaan kuuluvan käyttäjän oikeuksia täytyi järjestelmässä kyetä rajaamaan niin, että tietyille käyttäjille voitiin antaa suppeammat toimintamahdollisuudet kuin toisille. Esimerkkinä voidaan mainita ovimies, jonka käyttöoikeudet tulisi rajoittaa pelkästään kanta-asiakkaiden kirjaamiseen. Toisaalta taas ravintolapäälliköllä tulee olla täydet käyttöoikeudet kaikkiin järjestelmän toimintoihin asiakasraporteista uutisten ja asiakkaiden lisäämiseen. Käyttäjälle tulee siis kyetä määräämään oikeus kullekin järjestelmän toiminolle. Tästä syystä päädyittiin käyttämään käyttäjäoikeuksien ilmaisuun bittisarjaa, jonka jokainen numero vastaa tiettyä ennaltamäärättyä oikeutta. Numerosarjan kooksi määritettiin varmuuden vuoksi 32 bittiä, jotta tila varmasti riittäisi tulevaisuudenkin tarpeisiin. Kun jokainen bitti vastaa yhtä tiettyä oikeutta, mahdollisti 32 bittiä 32:n eri oikeuden määrämisen. Bitin arvo 0 eväsi käyttäjältä mahdollisuuden käyttää bitin määräämää toimintoa, kun arvo 1 antoi tähän luvan. Tekniikka mahdollisti yksityiskohtaisen käyttäjäoikeuksien määrittelyn ja tarkastuksen vain yhdellä 32 merkkiä pitkällä merkkijonolla, joka oli helppo säilöä tietokantaan.

Asiakaskuvien tallennukseen oli käytännössä kaksi vaihtoehtoa: joko normaali tiedostomuotoinen tallennus levylle tai kuvan muuntaminen binäärimuotoiseksi ja tallentaminen tietokantaan. Päädyimme jälkimmäiseen vaihtoehtoon. Näin pystyttiin helpommin yksilöimään asiakkaan kuva ja hänen tietonsa. Kuville ei myöskään tarvittu erillistä suojausta, koska tietokanta oli valmiiksi suojattu. Huonona puolena mainittakoon, että binäärimuotoiset kuvat vievät paljon tilaa tietokannassa ja saattavat näin hidastaa hakunopeutta. Tästä ei kuitenkaan syntyisi ongelmaa omassa järjestelmässä, koska kuvamäärät tulisivat olemaan maksimissaan tuhansia, jolloin hakuja hidastavaa vaikutusta ei vielä juurikaan näkyisi.

### 5.3.3Tietoturvakysymykset

Koska järjestelmä perustuu web-tekniikoihin ja sitä käytetään internetin yli web-selaimella, on tietoturvakysymykset otettava tavallista paikallista järjestelmää paremmin huomioon. Järjestelmämme keskeisimmät tietoturvatekijät ja uhat ovat tietokantainjektiot (nimenomaan SQL-kielelle ominaiset SQL-injektiot) ja käyttäjien salasanojen salassapito.

SQL-injektio on yleinen tapa hyökätä ja manipuloida tietoa tietokannassa. PHP-kielellä koodattuun järjestelmään voidaan syöttää GET- ja POST-metodeja käyttäen sisältöä, jolla skripteissä olevia SQL-lauseita voidaan yrittää muokata tietokannan väärinkäyttötarkoituksessa. Tällaiset hyökkäykset saadaan eliminoitua käyttämällä PHP-kielen valmista funktiota, jolla GET- ja POST-tiedot voidaan käsitellä turvallisiksi SQL-lauseita varten. Funktio mysql_real_escape_string on luotu juuri SQL-injektioita vastaan, ja nyrkkisääntönä voidaan sanoa, että jokainen GET- ja POST-metodilla vastaanotettu muuttuja on muokattava tällä funktiolla ennen muuttujan käyttämistä SQL-lauseissa. (PHP Security Guide 2005, kappale 3.2; PHP Manual 2008, mysql_real_escape_string -funktio)

Järjestelmä sisältää kaksi erilaista käyttäjän salasanaa. Ravintolahenkilökunnalle on omat tunnukset ja salasanat, sekä ravintolan asiakkaille asiakastunnukset ja salasanat. Kaikki salasanat on kryptattu tietokantaan sha1-kryptausmetodilla. Näin ollen kaikkien käyttäjien salasanat pysyvät heidän omana tietonaan, koska tietokanta sisältää vain kryptatun salasanan. Murtautuminen tietokantaan ei paljasta murtautujalle käyttäjien oikeita salasanoja, eikä tietokannan ylläpitäjäkään niitä tiedä. Sha1 on md5-kryptauksen kanssa yleisin tiedon kryptausmetodi PHP-sovelluksissa. (McGlinn 2005)

# 6 KEHITYSTYÖN TULOKSET

Kehitystyön tuloksena ja tuotteena syntyi sähköinen kanta-asiakasjärjestelmä, joka on suunnattu erityisesti nuoren asiakaskunnan omaaville ravintoloille ja yökerhoille. Järjestelmän ominaisuudet palvelevat niin ravintoloitsijaa kuin hänen asiakkaitaankin. Kehitystyö antoi tuotteen lisäksi myös kokemusta, tietoa ja taitoa tekijöilleen. Tietojärjestelmäprojekti on aina yksilöllinen tehtävä ja järjestelmäkehittäjän omaa ammattitaitoa ajatellen jokainen projekti antaa kehittäjälle uutta tietoa ja ratkaisuja uusiin ongelmiin. Seuraavaksi tarkastellaan kehitystyön tuloksia vastaamalla tutkimusasetelmassa asetettuihin tutkimuskysymyksiin.

## 6.1Järjestelmän toiminta ja hyöty ravintolalle ja sen asiakkaille

Järjestelmä koostuu kahdesta erillisestä osiosta: ravintolahenkilökunnalle tarkoitetusta hallintaosiosta ja ravintolan asiakkaille tarkoitetusta asiakasosiosta. Hallintaosiossa ravintola voi seurata asiakkaiden käyntikäyttäytymistä, informoida asiakkaita ravintolan toiminnasta sekä pitää kirjaa ravintolan tapahtumista ja kutsuvierastilaisuuksien vieras- ja seuralaislistoista. Käyntikäyttäytymisen seuraaminen on mahdollista, kun kanta-asiakkaat näyttävät kanta-asiakaskorttinsa ovella tullessaan sisään ravintolaan. Jokainen käynti tallentuu tietokantaan ja näin tiedetään jokaisen käynnin tarkka aika sekä asiakkaan sukupuoli. Järjestelmä ilmoittaa korttia näytettäessä kortinhaltijan tarkat tiedot sekä kuvan mikäli asiakkaan kuva on tallennettu tietokantaan. Kyseisestä toiminnosta on etua ravintoloitsijalle ja asiakkaille, koska vältytään kanta-asiakaskorttien väärinkäytöltä. Kun käyntejä tallennetaan tietokantaan, voi ravintoloitsija pyytää järjestelmältä raportin asiakkaiden käyntimääristä. Raportti ilmoittaa käyntimäärät eriteltynä kellonajoittain, päivän perusteella ja sukupuolen mukaan. Raporttitoiminto antaa tarpeellista tietoa, ja se onkin toiminto, jonka kehittämiseen kannattaa kiinnittää huomiota tulevaisuudessa.

Järjestelmä tarjoaa informaatiokanavan ravintolan ja sen asiakkaiden välille. Järjestelmän kautta kanta-asiakkaita voidaan informoida uutisten muodossa sekä ilmoittamalla erilaisista tapahtumista. Tapahtumien tietoihin voidaan myös lisätä mahdollisuus ilmoittaa seuralaiset, mikäli tapahtuma on seuralaistapahtuma. Tätä kautta seuralaisten

hanaikaisista excel-taulukoista päästään eroon, eikä niin sanotun avec-listan ylläpitämiseen kulu arvokkaita työtunteja. Lisäksi listan sähköinen ylläpitäminen tapahtuman aikana mahdollistaa sen, ettei kaikkien seuralaisten tarvitse saapua tapahtumaan kanta-asiakkaan mukana, vaan järjestelmä ilmoittaa kanta-asiakkaan perusteella kaikki kyseisen kanta-asiakkaan ilmoitetut seuralaiset ja sen ovatko he kirjautuneet sisään ravintolaan vai eivät.

Se, miten ravintolan kanta-asiakkaat hyötyvät järjestelmästä, on pitkälti kiinni ravintoloitsijasta, kuten aiemmin luvussa 4.2. todettiin. Asiakas hyötyy järjestelmästä helposti saatavilla olevan informaatiokanavan muodossa. Järjestelmän sivustolta asiakas löytää ravintolan toimintaan liittyvät uutiset ja tapahtumat ja voi näin hyödyntää kanta-asiakkuuttaan etenkin seuralaistapahtumissa. Lisäksi toteutettiin kaveritutka-niminen toiminto, jonka avulla kanta-asiakkaan on mahdollista seurata kanta-asiakkaana olevien ystäviensä käyntikäyttäytymistä ravintolassa. Kyseinen ystävä näkee web-sivustolta, mikäli hänen ystävänsä on mennyt sisälle yökerhoon, ja kenties näin käväistä siellä itsekin. Vaatimuksena siis on, että molemmat ovat yökerhon kanta-asiakkaita. Tällä toiminolla pyritään muodostamaan ryhmiä, joiden yökerhokäyttäytyminen keskittyisi juuri tähän nimenomaiseen yökerhoon. Tätä kautta kaveritutkasta hyötyy myös ravintoloitsija.

## 6.2Huomioonotettavat asiat tietojärjestelmää toteutettaessa

Tietojärjestelmää toteutettaessa on otettava huomioon paljon asioita. Projekti aloitetaan aina suunnittelulla, mutta suunnitteluvaihe ei liity pelkästään kehitystyön alkuun, vaan järjestelmää suunnitellaan läpi sen elinajan. Toteutusvaiheessa tulevat eteen tekniset kysymykset kuten käytettävä ohjelmointialusta, tiedon tallennustapa, tietoturva-asiat sekä kehitysympäristö. Toteutusvaiheen loppuun kuuluvat järjestelmän julkaisu ja käyttöönotto.

### 6.2.1Suunnittelu ja dokumentaatio

Tietojärjestelmäkehityksen tärkein vaihe, suunnittelu, sisältää suurimmaksi osaksi järjestelmän dokumentointia. Tämän työn dokumentointi alkoi vaatimusmäärittelyllä (liite 1), jossa mietittiin mitä järjestelmän käyttäjät halusivat järjestelmältä. Asiaa hahmoteltiin ravintolan asiakkaan ja ravintolapäällikön näkökulmista. Kun keskeisimmät

vaatimukset olivat selvillä, hahmoteltiin tärkeimmät käyttötapaukset ja luokat. Käyttötapauksia syntyi muutamia kymmeniä, joista keskeisimmät löytyvät liitteestä 3. Käyttötapauksien ja omien kokemuksien pohjalta rakennettiin käyttötapauskaaviot (liite 2), joissa ilmeni ensimmäistä kertaa järjestelmän toimintalogiikan pohja. Luokitusta mietittäessä alettiin hahmottamaan järjestelmän vaatiman tiedon rakennetta ja kyettiin luomaan alustava versio ER-kaaviosta (liite 4) sekä lopulta taulukuvaukset (liite 5).

Järjestelmän ohjelmallinen toteutusvaihe aloitettiin aikaisessa vaiheessa, ja dokumentaatiota päivitettiin toteutusvaiheen edetessä. Suunnitelmallisuudesta huolimatta voidaan todeta, että suunnittelua ei toteutettu täydellisesti, vaan muutamassa toiminnallisuudessa jouduttiin ottamaan askelia taaksepäin ja tekemään turhaa työtä. Suunnittelun osuus oli ajassa yli puolet koko järjestelmän kehityksestä. Näin ollen voidaankin todeta, että alan peukalosääntö, jonka mukaan suunnittelun osuus järjestelmän kehityksestä on noin 80%, pitää varsin hyvin paikkansa, mikäli järjestelmä halutaan toteuttaa tehokkaasti.

Kokonaisuutena voidaan kuitenkin todeta, että järjestelmän suunnitelu ja dokumentointi suoritettiin hyvin ja suuremmilta virheiltä vältyttiin. Järjestelmän dokumentaation tärkeimmät osat löytyvät liitteistä. Käyttötapauskaaviota, ER-kaaviota, taulukuvauksia seuraamalla on mahdollista ymmärtää suurimmaksi osaksi järjestelmän toimintaperiaate. Tulevaisuutta ajatellen dokumentaation hyötynä on sen käytettävyys uudelleenohjelmoinnissa. Mikäli järjestelmä halutaan siirtää uudelle ohjelmointialustalle, on suunnitteluvaiheen dokumentaatio käytännössä suurimmaksi osaksi valmis. Tietokantaan liittyvä dokumentaatio (ER-kaavio ja taulukuvaukset) ovat lähes suoraan käytettävissä, mikäli käytettävä tietokanta on SQL-standardin mukainen.

### 6.2.2Järjestelmän toteutus ja käyttöönotto

Järjestelmän toteuttamiseen tarvitaan ohjelmointi- sekä tietokantatyökalut sekä kehitysympäristö, joilla voidaan suorittaa järjestelmän koodin kirjoittaminen ja testaus suljetussa ympäristössä. Ohjelmallisen koodin kirjoittamisessa on tärkeää noudattaa käytettävän ohjelmointikielen tietoturvalliseksi todettuja metodeja tavallisimpiin toiminnallisuuksiin. Tietokantaa käytettäessä on suljettava pois erilaisten tietokantahyökkäysten mahdollisuus jo ohjelmakoodissa. Tärkeä seikka on myös miettiä julkaistavan version virheilmoitusten sisältö ja niiden käyttötilanteet. Julkaistava tuote tarvitsee lo-

puksi julkisen web-palvelimen ja tietokannan. Webhosting-palvelut tarjoavat edullisen tavan julkaista järjestelmä kohtuullisen vaivattomasti pienin kustannuksin.

Järjestelmän käyttöönotto tapahtuu rakentamalla tietokanta käyttäen valmiita luontilauseita sekä kopioimalla järjestelmän koodi julkiselle web-palvelimelle. Lisäksi tietokanta-asetukset tulee päivittää asetustiedostoon. Tietokantaan tulee lisätä pääkäyttäjä tarkoitusta varten luodulla tiedostolla. Tiedostoa muokkaamalla ennen sen ajamista, voidaan määrittää pääkäyttäjän tunnukset. Tiedosto ajetaan syöttämällä tiedoston url osoite riville, kun tietokanta on rakennettu. Tuoteversiosta on syytä poistaa kyseinen tiedosto sen ajamisen jälkeen väärinkäytösten välttämiseksi.

Kun pääkäyttäjä on lisätty, voidaan järjestelmän käyttöönottoa jatkaa itse ohjelman käyttöliittymästä. Pääkäyttäjän on mahdollista luoda uusia tunnuksia ravintolahenkilökunnalle, jotta järjestelmän keskeiset toiminnot saadaan aktiiviseen käyttöön. Henkilökunnan tunnusten oikeuksia suorittaa erilaisia toimintoja järjestelmän sisällä voidaan rajoittaa, ja tätä mahdollisuutta on syytä käyttää tarkasti, jotta vältytään järjestelmän väärinkäytöltä.

Jotta järjestelmä saadaan täysin toiminnalliseksi, on järjestelmään lisättävä lopuksi tunnukset myös asiakkaille. Asiakkaiden luominen on toteutettu niin, että asiakasta lisättäessä on asiakkaan tietojen lisäksi oltava kanta-asiakastunniste (magneettikortti tai viivakoodillinen kortti) ja kuva-asiakkaasta (ei pakollinen), jotka tallennetaan tietokantaan.

Kun järjestelmä on saatu käyttöön, alkaa kehitystyön tehneen projektiryhmän seuraava osuus, testaus todellisessa ympäristössä. Vaikka järjestelmää on testattu huolellisesti jo kehitysvaiheessa, ei koskaan voida tietää millaisia ongelmia suuret käyttäjämäärät ja erilaiset käyttäjäryhmät tuovat tullessaan. Tietojärjestelmän kehityskaari jatkuu koko järjestelmän eliniän ajan.

# 7 POHDINTA

Kehitystyön tulosten valossa voidaan todeta, että tietojenkäsittelyn tradenomin koulutus antaa tarvittavat tiedot ja taidot hyödyllisen ja tuotteistettavan tietojärjestelmän toteuttamiseksi ideoista aina toimivaksi järjestelmäksi asti. Tietojärjestelmät ovatkin tuotteita ja palveluita, joille on etenkin palvelualoilla kova kysyntä tulevaisuudessa. Yritykset tarvitsevat työkaluja etenkin asiakkuudenhallintaan, mutta tarpeita löytyy jokaiselta yritysmaailman osa-alueelta, jossa on käsiteltävää tietoa.

Tietojärjestelmä on tuote, jota on jatkuvasti kehitettävä. IT-ala kehittyy voimakkaasti uusien tekniikoiden ja teknologian myötä. Tämän projektin tuote on jo valmistuessaan jäänyt tietojärjestelmäkehityksen kirkkaimmasta kärjestä. Pärjätäkseen todellisilla markkinoilla järjestelmää on kehitettävä jatkuvasti. Projektiryhmän toiveissa on perustaa järjestelmälle oma yritys, joka kehittäisi ja ylläpitäisi sekä markkinoisi järjestelmää tulevaisuudessa. Kehitysideoita syntyi heti järjestelmän suunnitteluvaiheessa. Järjestelmän ideaa voidaan laajentaa monin eri tavoin. Toteutuskelpoisia ideoita olivat muun muassa mobiililaitteiden liittäminen järjestelmän toimintoihin sekä käyttöympäristön laajentaminen yhdestä ravintolasta usean ravintolan kanta-asiakasryhmien yhdistämiseksi.

Järjestelmää kehitettäessä heräsi idea järjestelmän laajentamisesta yleiseksi yömaailman tapahtuma- ja ravintolainformaatiopalveluksi ja www-sivustoksi. Järjestelmän asiakkaaksi päästyään voisi samaa korttia käyttää kaikissa ravintoloissa jotka ovat liittyneet palvelun käyttäjiksi. Näin ravintolat ja yökerhot saisivat paljon laajempaa tietoa ihmisten ravintolakäyttäytymisestä ja tätä kautta myös asiakkaat varmasti pääsisivät hyötymään uudenlaisista eduista. Lisäksi kaveritutkatoiminto kattaisi huomattavasti suuremman määrän mahdollisia sijainteja. Näin järjestelmää ei tarvitsisi erikseen pystyttää kullekin asiakkaalle vaan asiakasravintolan tulisi vain hankkia itselleen kortinlukulaite ja maksaa järjestelmän käyttämisestä esimerkiksi maksu per käynti -periaatteella. Tällöin voidaan puhua jo ennemmin nykyaikaisesta web-palvelusta kuin tuotteena myytävästä järjestelmästä. Nykyään web-pohjaiset tietojärjestelmät ovatkin muuttumassa räätälöidyistä tuotteista palveluiksi, joita järjestelmän kehittäjät tarjoavat.

Asiakkaille suunnattujen toimintojen lisäksi on syytä harkita, voitaisiinko järjestelmän käyttötarkoitusta laajentaa kattamaan myös ravintolan muuta toimintaa. Tämän työn luvussa 3.2. mainittiin laajennettu tarkoitus kirjainyhdistelmälle CRM, sisältäen kaikki ravintolan toimintaan osalliset osapuolet. Järjestelmään voitaisiin kehittää työkaluja myös henkilökunnan ja esiintyjien hallintaan. Nykysuuntaus tietojärjestelmissä onkin kattaa jonkin tietyn alan koko toiminta. Koko ravintola-alan (tarkentaen yökerhoalan) kattavan järjestelmän kehittäminen olisikin askel kohti Paul Hagenin (2006) artikkelissaan mainitsemaa osalliskeskeistä organisaatiota.

Kehitysnäkymiä suunniteltaessa on otettava huomioon myös järjestelmän toteuttamiseen käytettävät ohjelmointi- ja toteutustekniikat. PHP sopii hyvin pienemmille järjestelmille kuten tämän opinnäytetyön tuloksena syntyneelle järjestelmälle. Kun järjestelmää aletaan laajentaa ja sen käyttäjäkunta laajenee moninkertaiseksi, on syytä ottaa harkintaan uudelleenohjelmointi monipuolisemmalla ja enemmän nykypäivän vaatimuksia vastaavalla ohjelmointikielellä. Microsoftin DOT NET -arkkitehtuuri tekee voimakasta kasvua ohjelmistoalalla ja järjestelmän jatkokehitystä ajatellen se voisi olla hyvä vaihtoehto toteutusta varten. Myös vapaan lähdekoodin ohjelmointialustat on syytä ottaa ainakin harkinnan arvoisiksi vaihtoehdoiksi. Microsoftin alustaa voi pitää hyvänä vaihtoehtona siksi, että sen Live ID palvelut, joita edustavat parhaiten pikaviestin Live Messenger ja sähköpostipalvelu Hotmail, sisältävät valmiiksi valtavan asiakaskannan. Tätä asiakaskantaa voivat hyödyntää kaikki DOT NET -arkkitehtuuria hyödyntävät web-palvelut, joten kanta-asiakasjärjestelmästä voidaan tarvittaessa laajentaa kaikkien internetin käyttäjien ulottuvilla oleva web-palvelu. Tämä tulisi olla ainakin suuntana järjestelmän kehitystä ajatellen.

Olkoot järjestelmän kehityssuunta mikä tahansa, on varmaa että ravintola-alalla tullaan ottamaan käyttöön erilaisia web-pohjaisia järjestelmiä lähivuosina. Se on palvelualala siinä missä muutkin, eikä yhdelläkään alalla ole varaa jäädä kehityksestä jälkeen. Tämä opinnäytetyö on esimerkki siitä miten omista ideoista on mahdollista luoda käyttökelpoinen, toimiva ja hyödyllinen tietojärjestelmä. Tietojärjestelmäalalla tarvitaan jatkuvasti uusia ideoita ja niiden toteuttajia, koska olemassa olevaa tietoa, taitoa ja tekniikkaa käytetään vain murto-osa sen todellisesta potentiaalista. Liiketoimintaa ajatellen on edelleen kannattavaa kehittää web-pohjaisia järjestelmiä eri alojen yrityksille, etenkin pienille ja keskisuurille yrityksille. Järjestelmien toteuttaminen käy vuosi vuodelta helpommaksi kehitys- ja ohjelmointityökalujen kehityksen myötä. Oman jär-

jestelmän suunitteleminen ja toteuttaminen ei vaadi enää usean vuoden työtä ja riskisijoittajien rahoitusta. Etenkin palvelualoilla on suuri tarve erilaisen tiedon keräämiselle, tallentamiselle ja käsittelemiselle. Tätä tarvetta eivät tunnista yritykset itse, koska ne eivät ole tietoisia olemassa olevista mahdollisuuksista. Tietojenkäsittelyn ammatilaisille jää vastuu tarjota työkaluja, jotka helpottavat yritysten kuin niiden asiakkaidenkin jokapäiväistä elämää.

# LÄHTEET

Cato, J. 2001. User-centered web design. Harlow: Addison Wesley.

Heinisuo, R. 2003. PHP ja MySQL: tietokantapohjaiset verkkopalvelut. Helsinki: Talentum.

Hagen, P. 2006. Creating the relationship-centered organization: Non-profit CRM. Artikkeli asiakkuudenhallinnasta Idealware-sivustolla 5/2006. Viitattu 15.10.2006. http://www.idealware.org/articles/relationship_centric_org_CRM.php

Kaskela, L. 2005. Asiakkuudenhallinta ja sen merkitys. TIEKE:n Verkkokaveri.fi -sivustolla 21.6.2005. Viitattu 23.11.2006. http://www.tieke.fi/verkkokaveri/teemat/asiakkuuden_hallinta/asiakkuudenhallinta_ja_sen_merki/

Lehtonen, L. 2007. Tukipyyntö: PHP-ohjelmat. Sähköpostiviesti 1.3.2007. Vastaanottaja A. Vesterinen. Louhi.net web-hotellien asetukset.

Louhi Net. 2008. Webhotellien yleiset ominaisuudet. Louhi Net Oy:n kotisivuilta. Viitattu 12.10.2007. http://www.louhi.fi/?path=asiakaspalvelu

McGlinn, J. 2005. Password hashing. PHP Security Consortium -sivustolla 20.3.2005. Viitattu 27.9.2007. http://phpsec.org/articles/2005/password-hashing.html

Meltzer, M. 2004. CRM means change. The Wise Marketer -sivustolla 3/2004. Viitattu 15.10.2006. http://www.thewisemarketer.com/features/read.asp?id=43

PHP Online Manual. 2008. PHP-ohjelmointikielen sähköinen ohjekirja. Php.net-sivusto 2008. http://www.php.net

PHP Security Guide. 2005. Verkko-opas tietoturvallisen PHP-koodin kirjoittamiseen. Viitattu 24.2.2007. http://phpsec.org/projects/guide/

Vaswani, V. 2005. How to do everything with PHP & MySQL. Emeryville: McGraw-Hill/Osborne.

Vesterholm, M & Kyppö, J. 2006. Java-ohjelmointi. Jyväskylä: Gummerus.

Welling, L. & Thomson, L. 2001. PHP and MySQL Web Development. Indianapolis: Sams Publishing.

Wikipedia. 2008a. .NET. Wikipedia verkkotietosanakirjassa 8.1.2008. Viitattu 9.1.2008. http://fi.wikipedia.org/wiki/.net

Wikipedia. 2008b. Java. Wikipedia verkkotietosanakirjassa 3.1.2008. Viitattu 9.1.2008. http://fi.wikipedia.org/wiki/Java

# LIITTEET

## Liite 1. Järjestelmän toiminnalliset vaatimukset

1. Ylläpitäjän tulee kirjautua järjestelmään
2. Ylläpitäjiä voi lisätä ja poistaa
3. Ylläpitäjien oikeuksia voi muuttaa
4. Ylläpitäjien salasanoja voi muuttaa
5. Ylläpitäjä voi lisätä ja poistaa asiakkaita
6. Ylläpitäjä voi muuttaa asiakkaiden tietoja
7. Ylläpitäjä voi muuttaa asiakkaiden salasanoja
8. Ylläpitäjä voi kirjata asiakkaan sisään ja ulos ravintolasta tunnistautumisvälineen avulla
9. Ylläpitäjä voi lisätä, muokata ja poistaa uutisia
10. Ylläpitäjä voi lisätä, muokata ja poistaa tapahtumia
11. Ylläpitäjä voi kirjata sisään asiakkaan tunnistautumisvälineen avulla
12. Ylläpitäjä voi listata tapahtumaan ilmoitetut seuralaiset
13. Ylläpitäjä voi kirjata tapahtumaan ilmoitetun avecin sisään ravintolaan
14. Ylläpitäjä voi hyväksyä tai hylätä asiakkaan vaihtaman oman kuvan
15. Ylläpitäjä voi poistaa asiakkaan kuvan
16. Ylläpitäjä voi tulostaa raportin asiakkaiden käynneistä sukupuolen, viikonpäivän ja kellonajan mukaan eriteltynä
17. Ylläpitäjä voi kirjautua ulos järjestelmästä
18. Asiakkaan tulee kirjautua sisään järjestelmään

20. Asiakas voi muuttaa salasanansa

21. Asiakas voi vaihtaa kuvansa

22. Asiakas voi lähettää ystäväkutsun

23. Asiakas voi hyväksyä, hylätä tai torjua ystäväkutsun

24. Asiakas voi selata hyväksyttyjen ystäviensä käyntikäyttäytymistä

25. Asiakas voi selata tapahtumia

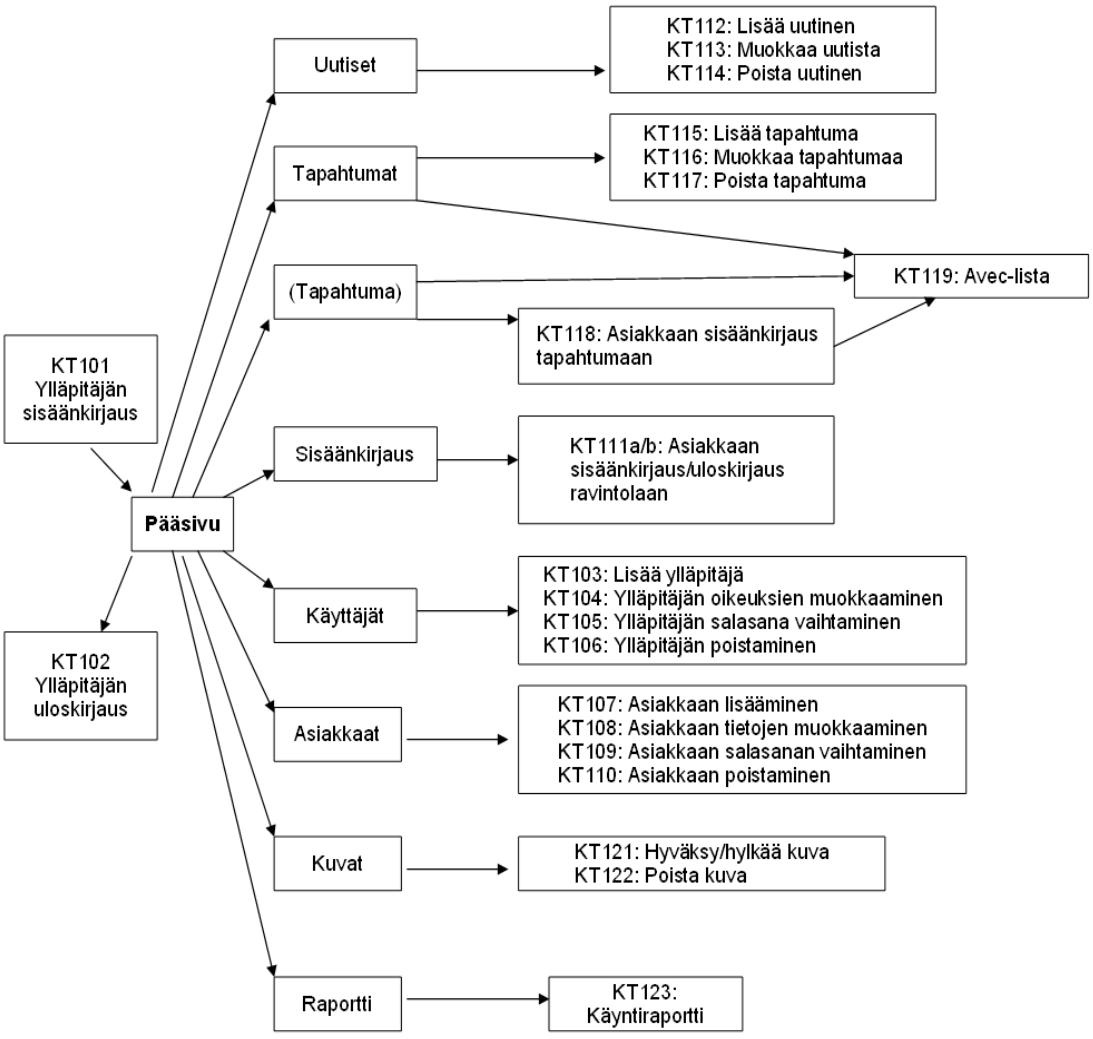
26. Asiakas voi ilmoittautua, ilmoittaa seuralaisensa, muokata ilmoittautumistaan tai poistaa ilmoittautumisensa tapahtumaan

27. Asiakas voi selata uutisia

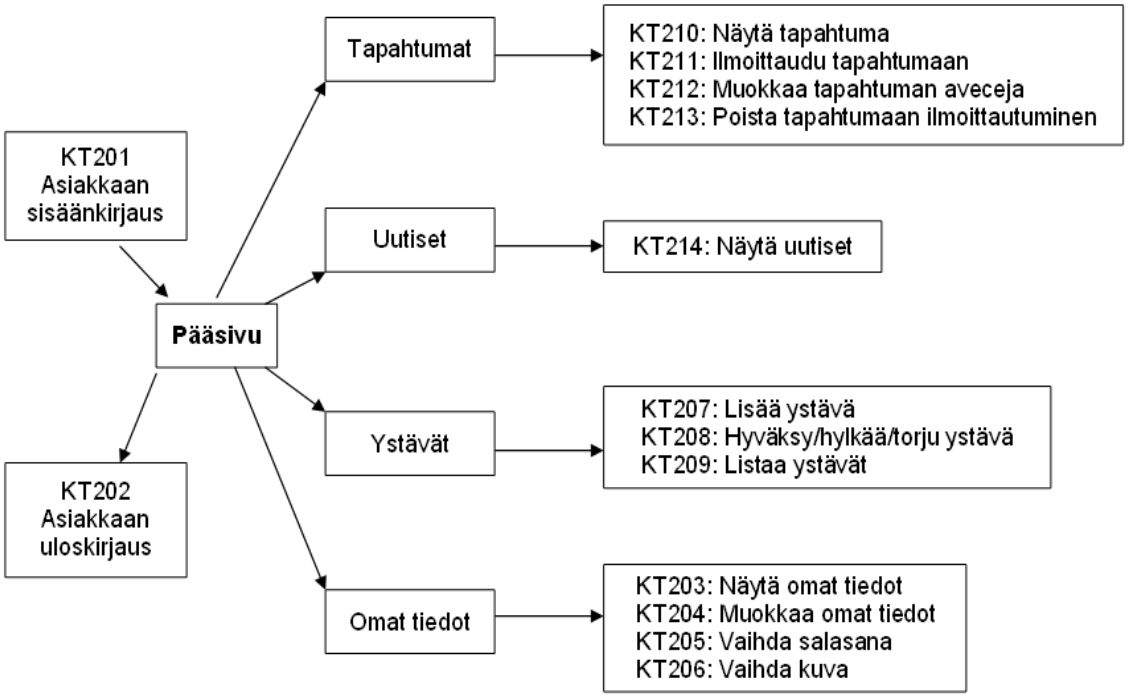
28. Asiakas voi kirjautua ulos järjestelmästä

# Liite 2. Käyttötapauskaaviot

Ylläpidon käyttötapauskaavio



Asiakkaan käyttötapauskaavio



# Liite 3. Käyttötapaukset

Ylläpidon käyttötapaukset KT101-KT123

Asiakkaan käyttötapaukset KT201-KT214

## KT101 Ylläpitäjän sisäänkirjaus

**Suorittaja** Ylläpitäjä

**Esiehdot** Ylläpitäjän tulee olla rekisteröitynyt järjestelmään.

**Kuvaus** Ylläpitäjä kirjautuu järjestelmään syöttämällä käyttäjätunnuksen ja tunnusta vastaavan salasanan..

**Poikkeukset** Mikäli annettu käyttäjätunnus ei ole olemassa, kirjautuminen ei onnistu ja annetaan virheilmoitus.

Mikäli annettu salasana ei täsmää annetun käyttäjätunnuksen kanssa, kirjautuminen ei onnistu ja annetaan virheilmoitus.

**Jälkiehdot** Ylläpitäjä on kirjautunut järjestelmään ja voi käyttää järjestelmän toimintoja käyttäjäoikeuksiensa rajoissa.

## KT102 Ylläpitäjän uloskirjaus

**Suorittaja** Ylläpitäjä

**Esiehdot** Ylläpitäjän tulee olla rekisteröitynyt järjestelmään ja kirjautunut sisään.

**Kuvaus** Ylläpitäjä kirjautuu ulos järjestelmästä klikkaamalla uloskirjauspainiketta.

**Poikkeukset** Mikäli käyttäjä ei ole kirjautunut sisään, ohjataan sisäänkirjaussivulle.

**Jälkiehdot** Ylläpitäjä on kirjautunut ulos järjestelmästä eikä voi käyttää järjestelmän toimintoja ennen sisäänkirjausta.

## KT103 Lisää ylläpitäjä

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus lisätä uusia ylläpitäjiä järjestelmään.

**Kuvaus** Ylläpitäjä lisää uuden ylläpitäjän järjestelmään antamalla halutun käyttäjätunnuksen ja salasanan kahdesti.

**Poikkeukset** Mikäli käyttäjätunnus on jo käytössä, ylläpitäjää ei lisätä ja annetaan virheilmoitus.

Mikäli annetut salasanat eivät täsmää, ylläpitäjää ei lisätä ja annetaan virheilmoitus.

**Jälkiehdot** Uuden ylläpitäjän käyttäjätunnus ja salasana tallentuu tietokantaan täysillä käyttäjäoikeuksilla.

## KT104 Ylläpitäjän oikeuksien muokkaaminen

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus muokata muiden ylläpitäjien oikeuksia.

Muokattava ylläpitäjä tulee olla rekisteröity järjestelmään.

**Kuvaus** Ylläpitäjä muokkaa toisen ylläpitäjän käyttöoikeuksia järjestelmään.

**Poikkeukset**

**Jälkiehdot** Muokattavan ylläpitäjän uudet oikeudet tallentuvat tietokantaan.

## KT105 Ylläpitäjän salasanan vaihtaminen

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus muokata muiden ylläpitäjien oikeuksia.

Muokattava ylläpitäjä tulee olla rekisteröity järjestelmään.

**Kuvaus** Ylläpitäjä vaihtaa toisen ylläpitäjän salasanan antamalla uuden salasanan kahdesti.

**Poikkeukset** Mikäli salasanat eivät täsmää, salasana ei vaihdu ja annetaan virheilmoitus.

**Jälkiehdot** Muokattavan ylläpitäjän uusi salasana tallentuu tietokantaan.

## KT106 Poista ylläpitäjä

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus poistaa ylläpitäjiä.

Poistettava ylläpitäjä tulee olla rekisteröity järjestelmään.

**Kuvaus** Ylläpitäjä poistaa toisen ylläpitäjän tunnukset järjestelmästä.

**Poikkeukset**

**Jälkiehdot** Poistetun ylläpitäjän tunnukset poistetaan tietokannasta

## KT107 Asiakkaan lisääminen järjestelmään

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus lisätä uusi asiakas järjestelmään.

Asiakkaan tiedot.

**Kuvaus** Ylläpitäjä lisää uuden asiakkaan järjestelmään syöttämällä asiakkaan tiedot.

**Poikkeukset** Mikäli asiakkaan käyttäjätunnus on jo olemassa, asiakasta ei lisätä ja annetaan virheilmoitus

Mikäli annetut salasanat eivät täsmää, asiakasta ei lisätä ja annetaan virheilmoitus.

Mikäli asiakkaan korttitunnus on jo rekisteröity toiselle asiakkaalle, asiakasta ei lisätä ja annetaan virheilmoitus.

**Jälkiehdot** Asiakkaan tiedot tallentuvat tietokantaan.

## KT108 Asiakkaan tietojen muokkaus

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus muokata asiakkaiden tietoja.

Muokattava asiakas tulee olla rekisteröity järjestelmään.

**Kuvaus** Ylläpitäjä muokkaa asiakkaan tietoja.

**Poikkeukset**

**Jälkiehdot** Asiakkaan muokatut tiedot päivittyvät tietokantaan.

## KT109 Asiakkaan salasanan vaihto

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus muokata asiakkaiden tietoja.

Asiakas tulee olla rekisteröity järjestelmään.

**Kuvaus** Ylläpitäjä vaihtaa asiakkaan salasanan.

**Poikkeukset** Mikäli annetut salasanat eivät täsmää, salasanaa ei vaihdeta ja annetaan virheilmoitus

**Jälkiehdot** Asiakkaan salasana päivittyy tietokantaan.

## KT110 Asiakkaan poistaminen

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus poistaa asiakkaita.

Poistettava asiakas tulee olla rekisteröity järjestelmään.

**Kuvaus** Ylläpitäjä poistaa asiakkaan tiedot järjestelmästä.

**Poikkeukset** Mikäli annetut salasanat eivät täsmää, salasanaa ei vaihdeta ja annetaan virheilmoitus

**Jälkiehdot** Asiakkaan tiedot poistetaan tietokannasta.

## KT111 Asiakkaan sisäänkirjaus/uloskirjaus ravintolaan

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus kirjata asiakkaita sisään ravintolaan.

**Kuvaus** Ylläpitäjä kirjaa asiakaan sisään ravintolaan (111a) tai ulos ravintolasta (111b).

**Poikkeukset** Mikäli annetun kortin tunnistetta ei ole rekisteröity, sisäänkirjaus ei onnistu ja annetaan virheilmoitus.

Mikäli asiakas on kirjattu sisään voidaan suorittaa vain uloskirjaus

Mikäli asiakas ei ole kirjautunut sisään ravintolaan, voidaan suorittaa vain sisäänkirjaus.

**Jälkiehdot** Käynnin tiedot tallennetaan tietokantaan.

## KT112 Uutisen lisääminen

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus lisätä uutisia.

**Kuvaus** Ylläpitäjä lisää uutisen järjestelmään.

**Poikkeukset**

**Jälkiehdot** Uutinen lisätään tietokantaan.

## KT113 Uutisen muokkaaminen

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus muokata uutisia.

Muokattava uutinen tulee olla järjestelmässä.

**Kuvaus** Ylläpitäjä muokkaa uutisen tietoja.

**Poikkeukset**

**Jälkiehdot** Uutisen tiedot päivitetään tietokantaan.

## KT114 Uutisen poistaminen

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus poistaa uutisia.

Poistettava uutinen tulee olla järjestelmässä.

**Kuvaus** Ylläpitäjä poistaa uutisen järjestelmästä.

**Poikkeukset**

**Jälkiehdot** Uutinen poistetaan tietokannasta.

## KT115 Tapahtuman lisääminen

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus lisätä tapahtumia.

**Kuvaus** Ylläpitäjä lisää tapahtuman tietoineen järjestelmään.

**Poikkeukset**

**Jälkiehdot** Tapahtuman tiedot tallennetaan tietokantaan.

## KT116 Tapahtuman muokkaaminen

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus lisätä tapahtumia.

Muokattava tapahtuma tulee olla tallennettuna järjestelmään.

**Kuvaus** Ylläpitäjä muokkaa tapahtuman tietoja.

**Poikkeukset**

**Jälkiehdot** Tapahtuman tiedot päivitetään tietokantaan.

## KT117 Tapahtuman poistaminen

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus poistaa tapahtumia.

Poistettava tapahtuma tulee olla järjestelmässä.

**Kuvaus** Ylläpitäjä poistaa tapahtuman järjestelmästä.

**Poikkeukset**

**Jälkiehdot** Tapahtuma poistetaan tietokannasta.

## KT118 Asiakkaan sisäänkirjaus tapahtumaan

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus kirjata asiakkaita tapahtumiin.

Asiakkaan tulee olla ilmoittautunut tapahtumaan.

Kirjausajankohta tulee olla tapahtumapäivänä tai korkeintaan 5h seuraavan vuorokauden puolella.

**Kuvaus** Ylläpitäjä kirjaa asiakkaan sisälle ravintolaan ilmoitettuun tapahtumaan.

**Poikkeukset**

**Jälkiehdot** Kirjautumisaika päivitetään asiakkaan tapahtumailmoittautumiseen (KT211).

## KT119 Avec-lista

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus selata avec-listoja.

**Kuvaus** Ylläpitäjä listaa halutun tapahtuman avec-tiedot.

**Poikkeukset** Mikäli tapahtumaan ei ole ilmoitettu yhtään seuralaista, näytetään tyhjä lista.

**Jälkiehdot** Avec-lista tulostetaan näytölle.

## KT120 Avecin kirjaaminen sisään tapahtumaan

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus kirjata aveceja sisään.

Asiakas on ilmoittanut avecin tapahtumaan.

Kirjausajankohta on oltava tapahtumapäivänä tai 5h seuraavan vuorokauden puolella.

**Kuvaus** Ylläpitäjä kirjaa sisään tulevan seuralaisen kirjautuneeksi tapahtumaan.

**Poikkeukset** Mikäli avec on jo kirjattu sisään, ei kirjautumista voida suorittaa.

**Jälkiehdot** Kirjautuminen päivitetään ilmoitetun avecin tietoihin (KT211) tietokantaan.

## KT121 Asiakkaan kuvan hyväksyminen

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus hyväksyä asiakkaiden kuvia.

Hyväksyttävä kuva tulee olla tallennettuna järjestelmään asiakkaan toimesta (KT206)

**Kuvaus** Ylläpitäjä hyväksyy asiakkaan kuvan, jonka asiakas haluaa itselleen vaihtaa (KT206).

**Poikkeukset**

**Jälkiehdot** Asiakkaan kuva vaihdetaan hyväksyttyyn kuvaan tietokantaan. Uusi kuva poistetaan hyväksyttävien kuvien listalta.

## KT122 Poista kuva

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus selata poistaa asiakkaiden kuvia.

Poistettava kuva tulee olla tallennettuna järjestelmään.

**Kuvaus** Ylläpitäjä poistaa halutun asiakkaan kuvan.

**Poikkeukset**

**Jälkiehdot** Asiakkaan kuva poistetaan tietokannasta.

## KT123 Käyntiraportti

**Suorittaja** Ylläpitäjä

**Esiehdot** Sisäänkirjautunut ylläpitäjä jolla oikeus selata käyntiraportteja.

**Kuvaus** Ylläpitäjä listaa halutun ajankohdan käyntiraportin.

**Poikkeukset**

**Jälkiehdot** Näytölle tulostetaan valitun ajankohdan asiakaskäynti määrät sukupuolittain, viikonpäivittäin ja puolen tunnin tarkkuudella.

## KT201 Kirjaudu sisään

**Suorittaja** Asiakas

**Esiehdot** Asiakkaan tulee olla rekisteröitynyt järjestelmään.

**Kuvaus** Asiakas kirjautuu järjestelmään syöttämällä käyttäjätunnuksen ja tunnusta vastaavan salasanan.

**Poikkeukset** Mikäli annettu käyttäjätunnus ei ole olemassa, kirjautuminen ei onnistu ja annetaan virheilmoitus.

Mikäli annettu salasana ei täsmää annetun käyttäjätunnuksen kanssa, kirjautuminen ei onnistu ja annetaan virheilmoitus.

**Jälkiehdot** Asiakas on kirjautunut järjestelmään ja voi käyttää järjestelmän toimintoja.

## KT202 Kirjaudu ulos

**Suorittaja** Asiakas

**Esiehdot** Asiakkaan tulee olla kirjautunut sisään.

**Kuvaus** Asiakas kirjautuu ulos järjestelmästä klikkaamalla uloskirjauspainiketta.

**Poikkeukset**

**Jälkiehdot** Asiakas on kirjautunut ulos järjestelmästä eikä voi käyttää järjestelmän toimintoja ennen sisäänkirjausta.

## KT203 Näytä omat tiedot

**Suorittaja** Asiakas

**Esiehdot** Sisäänkirjautunut asiakas.

**Kuvaus** Asiakas katselee omia tietojaan.

**Poikkeukset**

**Jälkiehdot**

## KT204 Muokkaa omat tiedot

**Suorittaja** Asiakas

**Esiehdot** Sisäänkirjautunut asiakas.

**Kuvaus** Asiakas muokkaa omia tietojaan.

**Poikkeukset** Tarvittava tieto puuttuu lomakkeelta.

**Jälkiehdot** Asiakkaan tiedot päivitetään tietokantaan.

## KT205 Vaihda salasana

**Suorittaja** Asiakas

**Esiehdot** Sisäänkirjautunut asiakas.

**Kuvaus** Asiakas vaihtaa salasanansa.

**Poikkeukset** Alkuperäinen salasana väärin.

Uudet salasanat eivät täsmää toisiinsa tai on liian lyhyt.

**Jälkiehdot** Salasana päivitetään tietokantaan.

## KT206 Vaihda kuva

**Suorittaja** Asiakas

**Esiehdot** Sisäänkirjautunut asiakas.

**Kuvaus** Asiakas lisää kuvan odottamaan ylläpitäjän hyväksyntää.

**Poikkeukset** Kuva väärän tyyppinen.

**Jälkiehdot** Kuva lisätty tietokantaan.

## KT207 Lisää ystävä

**Suorittaja** Asiakas

**Esiehdot** Sisäänkirjautunut asiakas.

**Kuvaus** Asiakas lähettää ystävä kutsun.

**Poikkeukset**

**Jälkiehdot** Kutsutulle ystävälle on lähetetty kutsu.

## KT208 Hyväksy/hylkää/torju ystäväksi

**Suorittaja** Asiakas

**Esiehdot** Sisäänkirjautunut asiakas.

Asiakas on saanut ystävä kutsun.

**Kuvaus** Asiakas hyväksyy, hylkää tai torjuu asiakkaan joka on lähettänyt hänelle ystävä kutsun.

**Poikkeukset**

**Jälkiehdot** Asiakkaiden välille syntyy tietokantaan yhteys.

## KT209 Tarkasta ystävät

**Suorittaja** Asiakas

**Esiehdot** Sisäänkirjautunut asiakas.

**Kuvaus** Asiakas tarkastaa onko ystäviä yökerhossa ja milloin viimeksi ovat olleet.

**Poikkeukset** Ei ole ollenkaan ystäviä.

**Jälkiehdot**

## KT210 Näytä tapahtuma

**Suorittaja** Asiakas

**Esiehdot** Sisäänkirjautunut asiakas.

**Kuvaus** Asiakas selailee tapahtumia.

**Poikkeukset** Ei ollenkaan tapahtumia.

**Jälkiehdot**

## KT211 Ilmottaudu tapahtumaan

**Suorittaja** Asiakas

**Esiehdot** Sisäänkirjautunut asiakas.

**Kuvaus** Asiakas ilmottautuu ja ilmoittaa seuralaiset tapahtumaan.

**Poikkeukset**

**Jälkiehdot** Asiakkaan ilmottautuminen ja seuralaiset lisätään tietokantaan.

## KT212 Muokkaa tapahtuman aveceja

**Suorittaja** Asiakas

**Esiehdot** Sisäänkirjautunut asiakas.

Ilmottautuminen tapahtumaan.

**Kuvaus** Asiakas muokkaa tapahtumaan ilmottamiaan seuralaisia.

**Poikkeukset**

**Kuvitus**

**Jälkiehdot** Seuralaisten tiedot päivitetään tietokantaan.

## KT213 Poista tapahtuman ilmoittautuminen

**Suorittaja** Asiakas

**Esiehdot** Sisäänkirjautunut asiakas.

Ilmottautuminen tapahtumaan.

**Kuvaus** Asiakas poistaa ilmottautumisensa tapahtumaan.

**Poikkeukset**

**Jälkiehdot** Ilmottautuminen tapahtumaan poistetaan tietokannasta.

## KT214 Näytä uutiset

**Suorittaja** Asiakas

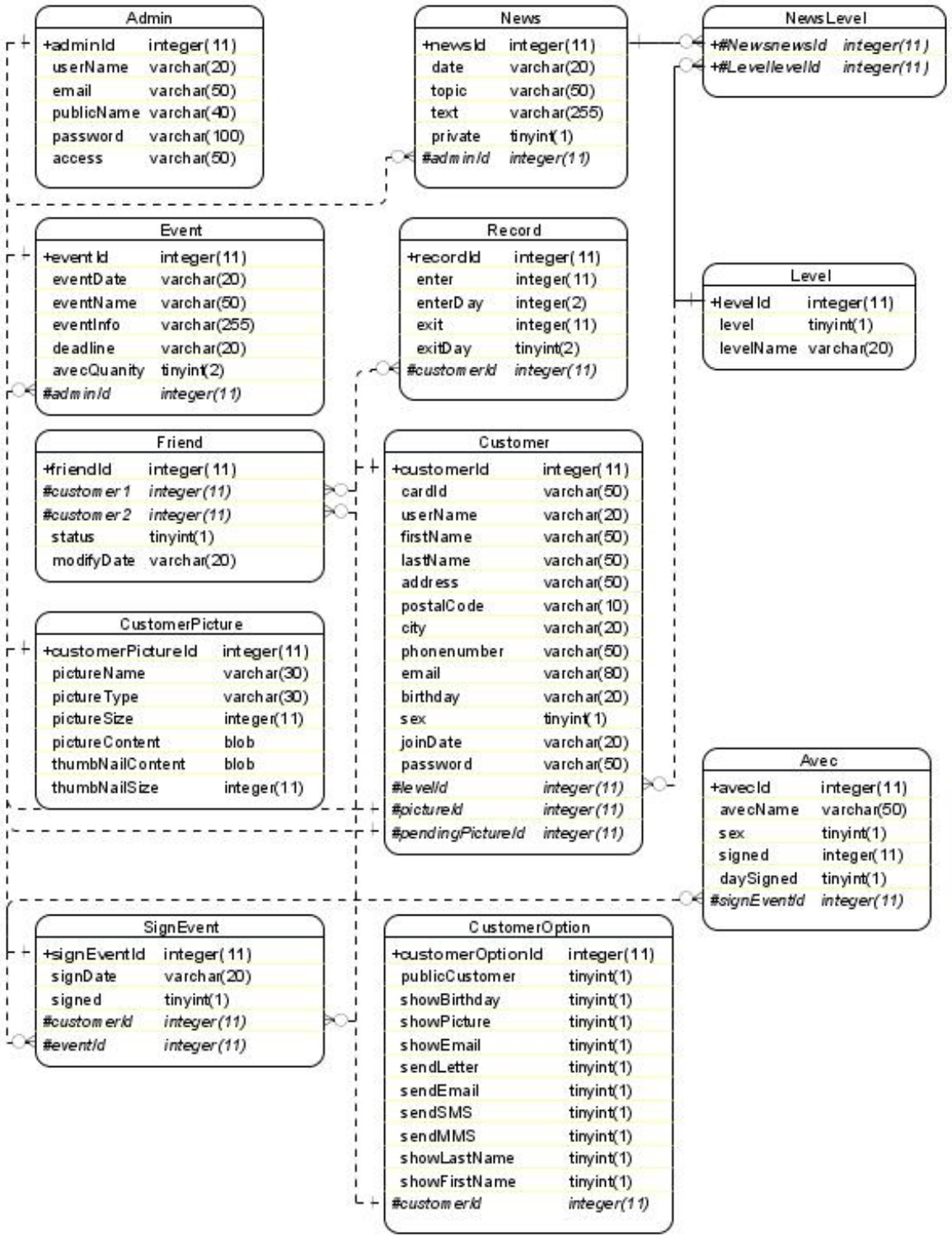
**Esiehdot** Sisäänkirjautunut asiakas.

**Kuvaus** Asiakas selailee uutisia.

**Poikkeukset** Ei ole uutisia.

**Jälkiehdot**

# Liite 4. ER-kaavio



# Liite 5. Taulukuvaukset

**ADMIN TAULU**

| Kenttä | Kuvaus | Null? | Tietotyyppi |
|---|---|---|---|
| adminId | Sisäinen id-numero. | NOT NULL | INTEGER(11) |
| userName | Käyttäjän tunnus. | NOT NULL | VARCHAR(20) |
| email | Käyttäjän sähköposti. | NOT NULL | VARCHAR(50) |
| publicName | Käyttäjän julkinen nimi. | NULL | VARCHAR(40) |
| password | Käyttäjän salasana. | NOT NULL | VARCHAR(100) |
| access | Käyttäjän oikeudet järjestelmään. | NOT NULL | VARCHAR(50) |

**Rajoitteet**

| Määritys | Tyyppi |
|---|---|
| PRIMARY KEY | adminId |

**AVEC TAULU**

| Kenttä | Kuvaus | Null? | Tietotyyppi |
|---|---|---|---|
| avecId | Sisäinen id-numero. | NOT NULL | INTEGER(11) |
| avecName | Seuralaisen nimi. | NOT NULL | VARCHAR(50) |
| sex | Seuralaisen sukupuoli. | NOT NULL | TINYINT(1) |
| signed | Tapahtuman sisäänkirjaus ajankohta. | NULL | INTEGER(11) |
| daySigned | Viikonpäivä. | NOT NULL | TINYINT(1) |
| signEventId | Viittaa tapahtuman tunnukseen. | NOT NULL | INTEGER(11) |

**Rajoitteet**

| Määritys | Tyyppi |
|---|---|
| PRIMARY KEY | avecId |
| FOREIGN KEY | signEventId for SignEvent (signEventId) |
| FOREIGN KEY | customerId for Customer (customerId) |

**CUSTOMER TAULU**

| Kenttä | Kuvaus | Null? | Tietotyyppi |
|---|---|---|---|
| customerId | Sisäinen id-numero. | NOT NULL | INTEGER(11) |
| cardId | Asiakkaan kortin numero. | NOT NULL | VARCHAR(50) |
| userName | Asiakkaan käyttäjätunnus. | NOT NULL | VARCHAR(20) |
| firstName | Asiakkaan etunimi. | NOT NULL | VARCHAR(50) |
| lastName | Asiakkaan sukunimi. | NOT NULL | VARCHAR(50) |
| address | Asiakkaan osoite. | NOT NULL | VARCHAR(50) |
| postalCode | Asiakkaan postinumero. | NOT NULL | VARCHAR(10) |
| city | Asiakkaan kaupunki. | NOT NULL | VARCHAR(20) |
| phonenumber | Asiakkaan puhelinnumero. | NOT NULL | VARCHAR(50) |
| email | Asiakkaan sähköposti osoite. | NOT NULL | VARCHAR(80) |
| birthday | Asiakkaan syntymäpäivä. | NOT NULL | VARCHAR(20) |
| sex | Asiakkaan sukupuoli. | NULL | TINYINT(1) |
| joinDate | Asiakkaan liittymis päivämäärä. | NOT NULL | VARCHAR(20) |
| password | Asiakkaan salasana salattuna. | NOT NULL | VARCHAR(50) |
| levelId | Viittaus taso tunnukseen. | NULL | INTEGER(11) |
| pictureId | Viittaus kuvan tunnukseen. | NULL | INTEGER(11) |
| pendingPictureId | Viittaus hyväksymättömän kuvan tunnukseen. | NULL | INTEGER(11) |

**Rajoitteet**

| Määritys | Tyyppi |
|---|---|
| PRIMARY KEY | customerId |
| FOREIGN KEY | levelId for Level (levelId) |
| FOREIGN KEY | pictureId for CustomerPicture (customerPictureId) |
| FOREIGN KEY | pendingPictureId for CustomerPicture (customerPictureId) |

**CUSTOMEROPTION TAULU**

| Kenttä | Kuvaus | Null? | Tietotyyppi |
|---|---|---|---|
| customerOptionId | Sisäinen id-numero. | NOT NULL | INTEGER(11) |
| publicCustomer | Näytetäänkö asiakas julkisesti. | NOT NULL | TINYINT(1) |
| showBirthday | Näytetäänkö syntymäpäivä. | NOT NULL | TINYINT(1) |
| showPicture | Näytetäänkö kuva. | NOT NULL | TINYINT(1) |
| showEmail | Näytetäänkö sähköposti. | NOT NULL | TINYINT(1) |
| sendLetter | Saako lähettää postia. | NOT NULL | TINYINT(1) |
| sendEmail | Saako lähettää sähköpostia. | NOT NULL | TINYINT(1) |
| sendSMS | Saako lähettää tekstiviestiä. | NOT NULL | TINYINT(1) |
| sendMMS | Saako lähettää multimediaviestejä. | NOT NULL | TINYINT(1) |
| showLastName | Näytetäänkö sukunimi. | NOT NULL | TINYINT(1) |
| showFirstName | Näytetäänkö etunimi. | NOT NULL | TINYINT(1) |
| customerId | Viittaus asiakkaan tunnukseen. | NOT NULL | INTEGER(11) |

**Rajoitteet**

| Määritys | Tyyppi |
|---|---|
| PRIMARY KEY | customerOptionId |
| FOREIGN KEY | customerId for Customer (customerId) |

**CUSTOMERPICTURE TAULU**

| Kenttä | Kuvaus | Null? | Tietotyyppi |
|---|---|---|---|
| customerPictureId | Sisäinen id-numero. | NOT NULL | INTEGER(11) |
| pictureName | Kuvan nimi. | NOT NULL | VARCHAR(30) |
| pictureType | Kuvan tiedostotyyppi. | NOT NULL | VARCHAR(30) |
| pictureSize | Kuvan koko. | NOT NULL | INTEGER(11) |
| pictureContent | Kuvan sisältö binäärinä. | NOT NULL | LONGBLOB |
| thumbNailContent | Pienoiskuvan sisältö binäärinä. | NOT NULL | LONGBLOB |
| thumbNailSize | Pienoiskuvan koko. | NOT NULL | INTEGER(11) |

**Rajoitteet**

| Määritys | Tyyppi |
|---|---|
| PRIMARY KEY | customerPictureId |

**EVENT TAULU**

| Kenttä | Kuvaus | Null? | Tietotyyppi |
|---|---|---|---|
| eventId | Sisäinen id-numero. | NOT NULL | INTEGER(11) |
| eventDate | Tapahtuman päivämäärä. | NOT NULL | VARCHAR(20) |
| eventName | Tapahtuman nimi. | NOT NULL | VARCHAR(50) |
| eventInfo | Tapahtuman kuvaus. | NULL | LONGTEXT |
| deadline | Tapahtumaan ilmottautumisen viimeinen päivä. | NULL | VARCHAR(20) |
| avecQuanity | Seuralaisten määrä. | NULL | TINYINT(2) |
| adminId | Viittaus tapahtuman ilmoittaneen käyttäjän tunnukseen. | NOT NULL | INTEGER(11) |

**Rajoitteet**

| Määritys | Tyyppi |
|---|---|
| PRIMARY KEY | eventId |
| FOREIGN KEY | adminId for Admin (adminId) |

**FRIEND TAULU**

| Kenttä | Kuvaus | Null? | Tietotyyppi |
|---|---|---|---|
| friendId | Sisäinen id-numero. | NOT NULL | INTEGER(11) |
| customer1 | Ystävä joka on tehnyt kutsun. | NOT NULL | INTEGER(11) |
| customer2 | Ystävä joka on saanut kutsun. | NOT NULL | INTEGER(11) |
| status | Ystävyyden tila. | NOT NULL | TINYINT(1) |
| modifyDate | Päivämäärä kun tila on muuttunut tai ystävyys luotu. | NOT NULL | VARCHAR(20) |

**Rajoitteet**

| Määritys | Tyyppi |
|---|---|
| PRIMARY KEY | friendId |

**LEVEL TAULU**

| Kenttä | Kuvaus | Null? | Tietotyyppi |
|---|---|---|---|
| levelId | Sisäinen id-numero. | NOT NULL | INTEGER(11) |
| level | Tason tyyppi. | NOT NULL | TINYINT(1) |
| levelName | Tason nimi. | NOT NULL | VARCHAR(20) |

**Rajoitteet**

| Määritys | Tyyppi |
|---|---|
| PRIMARY KEY | levelId |

**News taulu**

| Kenttä | Kuvaus | Null? | Tietotyyppi |
|---|---|---|---|
| newsId | Sisäinen id-numero. | NOT NULL | INTEGER(11) |
| date | Uutisen päivämäärä. | NOT NULL | VARCHAR(20) |
| topic | Uutisen otsikko. | NOT NULL | VARCHAR(50) |
| text | Uutisen teksti. | NULL | LONGTEXT |
| private | Uutisen näkyvyys taso. | NOT NULL | TINYINT(1) |
| adminId | Viittaus uutisen kirjoittaneen käyttäjän tunnukseen. | NOT NULL | INTEGER(11) |

**Rajoitteet**

| Määritys | Tyyppi |
|---|---|
| PRIMARY KEY | adminId |
| FOREIGN KEY | admin for Admin (adminId) |

**NEWSLEVEL TAULU**

| Kenttä | Kuvaus | Null? | Tietotyyppi |
|---|---|---|---|
| newsId | Sisäinen id-numero. | NOT NULL | INTEGER(11) |
| levelId | Kertoo uutisen tason. | NOT NULL | INTEGER(11) |

**Rajoitteet**

| Määritys | Tyyppi |
|---|---|
| PRIMARY KEY | newsId |
| FOREIGN KEY | levelId for Level (levelId) |

**RECORD TAULU**

| Kenttä | Kuvaus | Null? | Tietotyyppi |
|---|---|---|---|
| recordId | Sisäinen id-numero. | NOT NULL | INTEGER(11) |
| enter | Asiakkaan kirjautuminen. | NOT NULL | INTEGER(11) |
| enterDay | Kirjautumisen päivä. | NOT NULL | TINYINT(1) |
| exit | Asiakas uloskirjautuu. | NULL | INTEGER(11) |
| exitDay | Uloskirjautumisen päivä. | NULL | TINYINT(1) |
| customerId | Viittaus asiakkaan tunnukseen. | NOT NULL | INTEGER(11) |

**Rajoitteet**

| Määritys | Tyyppi |
|---|---|
| PRIMARY KEY | recordId |
| FOREIGN KEY | customer for Customer (customerId) |

**SIGNEVENT TAULU**

| Kenttä | Kuvaus | Null? | Tietotyyppi |
|---|---|---|---|
| signEventId | Sisäinen id-numero. | NOT NULL | INTEGER(11) |
| signDate | Tapahtuman kirjauksen päivämäärä. | NOT NULL | VARCHAR(20) |
| signed | Kirjauksen tila. | NOT NULL | TINYINT(1) |
| customerId | Viittaus asiakkaan tunnukseen. | NOT NULL | INTEGER(11) |
| eventId | Viittaus tapahtuman tunnukseen. | NOT NULL | INTEGER(11) |

**Rajoitteet**

| Määritys | Tyyppi |
|---|---|
| PRIMARY KEY | signEventId |
| FOREIGN KEY | eventId for Event (eventId) |