VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Patrik Bexar

NEXTCALL PRO LIVESCREEN - NÄYTTÖSOVELLUS

Tekniikka ja liikenne
2011

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Patrik Bexar
Opinnäytetyön nimi	NextCall Pro Livescreen -näyttösovellus
Vuosi	2011
Kieli	suomi
Sivumäärä	53
Ohjaaja	Pirjo Prosi

Opinnäytetyön tarkoituksena on tehdä kevyempi, helpommin käytettävä ja selkeä näyttösovellus puhelutietojen reaaliaikaiseen näyttämiseen Anvian valvomo- ja helpdesk-organisaatiossa sekä raportointiosa, jolla saadaan haettua tilasto- ja historiatietoja yhteydenotoista ja vastaajista eli agenteista. Näyttö tullaan näyttämään valvomossa ja helpdeskissä suurilta näytöiltä, jolloin tilannetta on helppo seurata ja reagoida, mikäli jossain tarvitaan lisäresursseja.

Sovellukset toteutettiin PHP-ohjelmointikielellä HTML:ä, CSS:ä ja jQuery JavaScript-kirjastoa apuna käyttäen. Kaikki sovelluksissa käytettävät tiedot haetaan työn pohjana olevan NextCall Pro-sovelluksen PostgreSQL-tietokannasta. Tietokantakyselyiden tulee olla mahdollisimman kevyitä ja tehokkaita, jotta jo valmiiksi suuren rasituksen alla oleva NextCall Pro-sovelluksen tietokanta ei kuormitu liikaa. Tässä työssä on kiinnitetty erityisesti huomiota tietokantakyselyiden tehostamiseen ja sovellusten tietoturvallisuuteen. Internetpalveluntarjoajana Anvia on tärkeässä osassa tietoturvallisuuden saralla, joten on tärkeää, että sovellukset, jotka sijaitsevat heidän palvelimillaan ovat tietoturvallisesti toteutettuja. Tietoturvallisuuden osalta tässä työssä huomioidaan sekä turvallinen tietokannan käsittely että PHP-koodin tietoturvallisuus.

Opinnäytetyön tuloksena toteutettiin kaksi tietoturvallista, suorituskykyistä ja helposti laajennettavaa sovellusta.

Avainsanat PHP, PostgreSQL, tietoturvallisuus, tietokanta, tehokkuus

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikan koulutusohjelma

ABSTRACT

Author	Patrik Bexar
Title	NextCall Pro Livescreen application
Year	2011
Language	Finnish
Pages	53
Name of Supervisor	Pirjo Prosi

The purpose of this thesis work was to develop a lighter, easier to use and more explicit livescreen application to show real time data of incoming phone calls for Anvia valvomo- helpdesk-organisation, in addition to a reporting application which can be used to get historic or statistic information about incoming events like phone calls and emails and about agents that answers to these events. The livescreen will be shown in Anvia monitoring room and HelpDesk on two big screens. Therefore it will be easy to react on if more resources are needed somewhere.

The applications were made by using PHP programming language with help of HTML, CSS and jQuery JavaScript libraries. All the data used in the applications are fetched from NextCall Pro-applications PostgreSQL-database that is the base for this thesis work. The SQL-queries had to be as light and effective as possible to avoid stressing the already stressed PostgreSQL-database of NextCall Pro-application. The thesis work observes closely making the SQL-queries as effective as possible. Another thing that got special attention in this thesis was to make the applications as secure as possible. Anvia is an internet service provider and therefore very concerned about internet security and specially interested that the applications on their servers are secure. Both secure database handling and secure PHP-programming are taken into consideration.

As a result of the thesis work two secure, effective and easily expandable applications was created.

Keywords PHP, PostgreSQL, security, database, efficiency

SISÄLLYS

1 JOHDANTO

1.1 Työn tarkoitus

Tämä työ pohjautuu Capricode:n tekemään NextCall Pro-sovellukseen, joka on Anvialla käytössä oleva puhelunohjausjärjestelmä. Puhelinohjausjärjestelmä koostuu kahdesta osasta: puhelinkeskuksen yhteydessä toimivasta palvelinosasta, joka ohjaa tulevia puheluita ja sähköposteja eri sarjoihin ja edelleen agenteille, ja agenttisovelluksista, joita ajetaan käyttäjien koneilla. Käyttäjä kirjautuu sisään agenttisovellukseen ja valitsee mistä sarjoista hän ottaa tapahtumia vastaan ja sen perusteella palvelinsovellus osaa ohjata niiden sarjojen tapahtumia agentille. Agentti on käyttäjän koneella ajettava sovellus, johon tapahtumat eli puhelut ja sähköpostit ohjataan. Agenttisovellukseen kirjauduttaessa sovellukselle kerrotaan puhelinnumero johon puhelut ohjataan sekä mistä sarjoista agentti ottaa tapahtumia vastaan. Kaikki tapahtumat tallennetaan palvelimella sijaitsevaan PotgreSQL-tietokantaa. Tietokantaan tallennetaan tieto myös, kun agentti kirjautuu sisään, menee tauolle, vaihtaa sarjoja tai sulkee sovelluksen. NextCall Pro:n Agenttisovellus näyttää käyttäjälle montako tapahtumaa jonottaa hänen valitsemissaan sarjoissa, sekä tiedon vapaana olevista agenteista. Sovellus ei erittele valittujen sarjojen jonoja, vaan näyttää, montako puhelua sarjoissa on jonossa yhteensä.

Tämän työn tarkoituksena on tehdä kevyempi helpommin käytettävä ja selkeä näyttösovellus puhelutietojen reaaliaikaiseen näyttämiseen Anvian valvomo- ja helpdesk-organisaatiossa. Sovellus näyttää sarjojen sen hetkiset jonomäärät, jonotusajat, vastausprosentin, puhelumäärän kuluvalta vuorokaudelta sekä tiedon montako agenttia on kirjautuneena kuhunkin sarjaan. Näyttö tullaan näyttämään valvomossa ja helpdeskissä suurilta näytöiltä, jolloin tilannetta on helppo seurata ja reagoida, mikäli jossain sarjassa tarvitaan lisäresursseja. Tilanteen seuraamisen helpottamiseksi sovellukseen voidaan syöttää raja-arvoja, joiden alittuessa tai ylittyessä sovellus näyttää kyseisessä kentässä varoituksen. Lisänä toteutetaan raportointisovellus, jolla voidaan hakea erilaisia tietoja ja tilastoja helpon käyttöliitty-

män kautta. Sovellus tehdään Anvian valvomo- ja helpdesk-organisaatiolle, mutta se tullaan ottamaan käyttöön myös Anvian asiakaspalvelussa talven 2011 aikana.

Internetpalvelut ovat nykyään tärkeä osa ihmisten jokapäiväistä elämää ja kilpailun koventuessa internetpalveluntarjoajien kesken asiakaspalvelun, vikailmoitusten sekä helpdesk-palvelujen nopeat vasteajat ovat huomattava kilpailuetu operaattoreille. Tästä syystä Anvialla halutaan jatkuvasti kehittää näitä palveluita. Tämän opinnäytetyön tuloksena tehdyn sovelluksen tarkoituksena on helpottaa näiden palveluiden sekä reaaliaikaisten että tilastollisten tietojen seurantaa. Tietoa puheluiden määristä ja vastausprosenteista lähetetään kuukausittain sekä Anvian johdolle että viestintävirastolle. Tämän sovelluksen ansiosta näitä tietoja voidaan seurata reaaliajassa ja tiedot saadaan helposti ja kätevästi haettua halutussa muodossa.

Anvialle on tärkeää, että sovellukset jotka sijaitsevat heidän palvelimillaan ovat tietoturvallisesti toteutettuja. Tässä työssä on erityisesti kiinnitetty huomiota sovelluksen tietoturvallisuuteen vaikka sovellukset, jotka tämän työn tuloksena toteutettiin, sijaitsevat Anvian palomuurien sisällä ja niihin pääsee käsiksi vain Anvian sisäverkosta. Tietoturvan ollessa kyseessä ei voi koskaan olla liian varovainen.

Näyttösovelluksessa käytetty tietokanta on käytössä NextCall Pro-sovelluksessa ja jo valmiiksi suuren rasituksen alla. Tästä syystä tätä työtä tehtäessä kiinnitettiin erityistä huomiota tietokantakyselyiden tehostamiseen ja tietokannan mahdollisimman pieneen rasittamiseen.

1.2 Toimeksiantajat

Tämä työ tehtiin Anvia-konserniin kuuluvan Telecom-liiketoiminta-alueen valvomo- ja helpdesk-organisaation tarpeisiin. Mukana oli myös Telecom-liiketoiminta-alueella käytössä olevan puhelinliikenteenjakajan Anvialle tuottanut yritys Capricode.

1.2.1 Anvia

Anvia, alkujaan Waasan Telefooniyhdistys, oli ensimmäisten joukossa Suomessa tarjoamassa puhelinpalveluja Vaasan yrityksille ja asukkaille. Ensimmäiset puhelimet tulivat käyttöön Vaasassa lokakuussa 1882 vain kuusi vuotta Alexander Graham Bellin patentoitua keksimänsä puhelimen. Vaasa oli viides kaupunki Suomessa, johon perustettiin puhelinyhtiö Oy Waasan Telefooniyhdistys. Yhdistystä perustamassa olivat mm. kauppias Axel Schauman, myllynjohtaja Hugo Sölfverarm, kauppias Alfred Hedman sekä muita aktiivisia. Vuonna 1893 perustettiin myös Etelä-Pohjanmaalle oma puhelinyhtiö Etelä-Pohjanmaan Puhelin Oy. Nämä kaksi yhtiötä fuusioituivat vuonna 1989 Vaasan Läänin Puhelin Oy:ksi. Elokuussa 2008 yhtiö vaihtoi nimekseen Anvia. Uudistuneen strategian ja uuden monialaisemman konsernin nimeksi sopi paremmin Anvia, joka kuvaa yhteyttä ja eteenpäin menemistä. /3/

Anvia-konserniin kuuluu neljä eri liiketoiminta-aluetta: Telecom eli tietoliikenteen palvelu- ja verkkoliiketoiminta, Securi-turvaliiketoiminta, IT-liiketoiminta sekä TV-liiketoiminta. Tämä työ on tehty nimenomaan Telecom-liiketoiminta-alueelle. Telecom-liiketoiminta-alue toimii lähinnä Rannikko-Pohjanmaalla ulottuen Seinäjoen itäpuolelle Kokkolan ja Kristiinankaupungin välisellä alueella. Telecom liiketoiminta-alue keskittyy ns. yhtiön vanhoille liiketoiminta-alueille eli puhelin-, televisio- ja internet-yhteyksien ja palvelujen tuottamiseen. Anvialla työskentelee tällä hetkellä n. 700 työntekijää. Vuoden 2009 liikevaihto oli 103 M€. /2/

1.2.2 Capricode

Capricode on vuonna 2002 perustettu ohjelmistotalo. Capricoden pääkonttori sijaitsee Oulussa. Myyntikonttoreita yrityksellä on sekä Helsingissä että Lontoossa. Lisäksi heillä on ohjelmointikonttori Budapestissa. Capricode on erikoistunut telekommunikaatio-ohjelmistojen tuottamiseen. Yrityksen tärkeimpiä tuotteita ovat SyncShield®, joka on mobiililaitteiden hallintasovellus sekä erilaiset tietokoneen ja puhelinjärjestelmiä yhdistävät sovellukset kuten tämän työn pohjana käytetty NextCall Pro -puhelinliikenteen ohjausjärjestelmä. /5/

2 KÄYTETYT MENETELMÄT

Työ päätettiin toteuttaa PHP-ohjelmointikieltä käyttäen. PHP:n rinnalla käytettiin myös JavaScriptiä ja etenkin sen johdannaisia AJAX:a ja jQueryä. Tietokantana toimi Anvialla käytössä olevan puhelinliikenteenjakajan NextCall Pro:n PostgreSQL-tietokanta. Tässä kappaleessa perehdytään lyhyesti opinnäytetyössä käytettyihin menetelmiin sekä niiden taustoihin.

2.1 PHP

PHP on laajalti käytössä oleva palvelinpuolen skriptikieli. Se on käytössä yli 22 miljoonalla domain-sivustolla ja yli 1,4 miljoonalla palvelimella. PHP:n suosio perustuu pitkälti sen hyviin ominaisuuksiin kuten yksinkertaisuuteen, hyvään suorituskykyyn, jatkuvasti kehittyviin ominaisuuksiin sekä hyviin tietokantayhteysmahdollisuuksiin ja oliomaisen ohjelmoinnin tukeen uudemmissa versioissa. /1/

PHP, joka on rekursiivinen lyhenne sanoista HypertexText Preprosecessor, on alunperin Rasmus Lendorfin vuonna 1994 kehittämä skriptikieli palvelinpuolen dynaamisten web-sovellusten luontiin. PHP oli tuolloin vielä vain joukko julkaisumakroja, joilla pystyttiin ylläpitämään kotisivuja. Lendorf käytti ensimmäisestä versiosta nimitystä ”Personal Home Page Tools”. Toinen versio oli nimeltään PHP/FI ”Personal Home Page / Forms Interpreter”. Vasta version PHP/FI 2.0 jälkeen siirryttiin käyttämään yleisesti lyhennettä PHP. /16; 23/

PHP 3, joka on ensimmäinen nykyistä PHP:ta muistuttava versio, sai alkunsa, kun kaksi israelilaista sovelluskehittäjää Andi Gutmans ja Zeev Suraski vuonna 1997 päättivät kirjoittaa edellisen PHP-tulkin lähdekoodin täysin uusiksi saadakseen PHP:stä paremmin Internet-kauppasovellusten tarpeisiin sopivaksi. He ottivat yhteyttä Rasmus Lendorfiin ja yhdessä kolmikko päätti kehittää PHP/FI:n pohjalta täysin uuden itsenäisen ohjelmointikielen. Ohjelmointikieli sai uudeksi nimekseen pelkän PHP. Heidän versionsa, josta käytetään nimitystä PHP 3.0, suurimpia vahvuuksia oli sen vahva laajennettavuus. PHP 3.0 tarjosi hyvät rajapinnat useimmille tietokannoille ja protokollille sekä mahdollisti oliomaisen ohjelmoinnin ja tar-

josi huomattavasti tehokkaamman ja yhtenäisemmän syntaksin kuin edeltäjänsä. Toisia kehittäjiä projektiin mukaan houkutteli itse ohjelmointikielen helppo laajennettavuus. Myöhemmin Andi Gutmans ja Zeev Suraski perustivat yrityksen nimeltä Zend Technologies, joka on vielä tänäkin päivänä tärkeä tekijä PHP:n kehityksessä. Gutmansin ja Suraskin kehittämä Zend Engine on PHP 4:n ydin. Päämääränä heillä oli parantaa PHP 3.0:ssa esiteltyjen ominaisuuksien tehokkuutta, jotta ne vastaisivat vaativienkin sovellusten tarpeita. Tämän lisäksi PHP 4.0 toi mukanaan HTTP-sessionit, tulostuspuskurit, useita turvallisuus parannuksia sekä useita muita ominaisuuksia. Vuonna 2004 julkaistussa PHP 5.0:ssa on ytimenä uusi Zend Engine 2, joka mahdollistaa täysiverisen olio-ohjelmointimaisuuden PHP:ssä sekä parantaa PHP:n tehokkuutta ja turvallisuutta. /19; 22; 24/

PHP on avointa lähdekoodia (Open Source) eli kuka tahansa voi maksutta käyttää ja asentaa sen erilaisille alustoille. Lisäksi PHP:n lähdekoodia voi tarvittaessa muokata ja kopioida PHP License v3.01 puitteissa. PHP-lisenssi on BSD-tyylinen lisenssi. PHP:ta ylläpitää ja julkaisee PHP Community. Tällä hetkellä uusin versio on 5.3.8. /21/

PHP on palvelimella suoritettava ohjelmointikieli, joka on upotettuna html-dokumenttien sisään. PHP-ohjelma on tavallinen tekstimuotoinen lähdekooditiedosto, jonka päätteenä on .php. Pelkkä .php-tiedosto ei vielä tee mitään vaan se täytyy ensin tulkata. Tämä tapahtuu palvelimella ohjelmaa suoritettaessa. Palvelin huomaa pyynnössä esiintyvästä tarkenteesta eli .php-päätteestä, että kyseessä on tulkkausta vaativa PHP-koodia sisältävä dokumentti. PHP-kielen aloitus- ja lopetustunnisteiden sisällä oleva ohjelmakoodi annetaan tällöin Zendin ajonaikaiselle kääntäjälle. Kun kääntäjä on kääntänyt koodin suorittimen ymmärtämään muotoon, se annetaan suorittimelle, joka suorittaa itse koodin. Kun koodi on suoritettu, palautetaan suorittimen luoma HTML-koodi asiakassovellukselle, eli käytännössä selaimelle, joka näyttää sen käyttäjälle. PHP on siis ns. tulkkaukseen perustuva kieli, joka tulkataan jokaisen suorituksen yhteydessä uudestaan. PHP:n syntaksi on suurelta osin johdettu pearlista ja c-kielestä. PHP-koodi kirjoitetaan ylei-

sesti HTML-koodin sekaan ja se tunnistetaan tulkkausvaiheessa PHP-koodiksi aloitus- ja lopetusmerkkien avulla. Aloitusmerkki <?php ja lopetusmerkki ?>

```
<html>
    <head>
        <title>Esimerkki</title>
    </head>
    <body>
        <?php
            echo "Terve maailma!";
        ?>
    </body>
</html>
```

Kuva 1. Yksinkertainen PHP sovellus.

PHP:llä, toisin kuin pelkällä HTML:llä, pystytään helposti tekemään vuorovaikutteisia sovelluksia. PHP on nykyään lähes täysiverinen olio-ohjelmointikieli viimeisimpien versioiden myötä tulleiden olio-ominaisuuksien ja virheenkäsittelymekanismien myötä. /17/

2.2 HTML

HTML, ja käytännössä myös koko www (World Wide Web), sai alkunsa vuonna 1989, kun CERN:ssä töissä ollut Tim Berners-Lee kehitti järjestelmän, jolla pystyttiin jakamaan tutkimustuloksia, kuvia ja muuta tietoa verkon yli kaukaisistakin tutkimuslaitoksista. Pelkän dokumenttien koneelta koneelle siirtämisen sijaan, hänen ehdotuksessaan oli mahdollista linkittää tekstiin tiedostojen sisällä. Tämä tarkoitti sitä, että olisi mahdollista yhtä dokumenttia luettaessa nopeasti hypätä toisessa tutkimuksessa olevaan tietoon nappia painamalla. Näin ollen kaikki tutkimustieto olisi mahdollista linkittää yhdeksi suureksi verkostoksi tietoa. Ennen CERN:iin tuloa Berners-Lee oli työskennellyt tiedoston muodostuksen ja tekstin prosessoinnin kanssa ja kehittänyt omaan käyttöön tarkoitetun hyperteksti systeemin, jota hän käytti HTML:n perustan pohjana. Hypertekstillä tarkoitetaan tekstiä,

johon voidaan esimerkiksi upottaa linkkejä toisille sivuille, kuvia tai tiedostoja. Tällä tavalla tietoa haettaessa ei enää tarvinnut tietää koneesta, josta tieto haettiin, muuta kuin URL-osoite. /9; 11/

HTML eli HyperText Markup Language on www-sivujen rakennetta kuvaava kieli. Käytännössä HTML vain kertoo, kuinka jokin teksti selaimessa näytetään, esimerkiksi mikä osa tekstiä on otsikko tai mikä kohta on linkki ja minne se osoittaa. Aikaisemmin HTML:ä käytettiin lähes yksinään verkkosivujen tekoon mutta nykyään sitä käytetään lähinnä vain muiden kielten, kuten PHP:n tulosten näyttämiseen ja muotoiluun. /9/

HTML-koodi rakentuu elementeistä eli tagien sisään tulevasta sisällöstä. Elementeillä on kaksi perusominaisuutta, attribuutit ja itse sisältö. Toisin kuin esimerkiksi XML:ssä, HTML:ssä kaikki käytettävät elementit ovat valmiiksi nimettyjä. Elementit koostuvat aloitetunnisteesta ja lopetustunnisteesta. Elementtien attribuutit sijaitsevat aloitustunnisteessa. Lopetustunniste on samanlainen kuin aloitustunniste, mutta ilman attribuutteja. Aloitustunniste voi sisältää myös / -merkin, joka kertoo, että se on lopetustunniste. HTML-tunnisteiden väliin tulee itse elementin sisältö, esimerkiksi näytettävä teksti. HTML:ssä kaikilla elementeillä, joilla on sisältö, pitää olla sekä aloitus- että lopetustunniste. Esimerkki sisällöttömästä elementistä on rivinvaihtoelementti
. /9/

HTML-elementtejä on kolmea eri tyyppiä. Nämä tyypit ryhmitellään sen mukaan, minkälaista merkkausta ne kuvaavat. Yksi on esitykselliset elementit, jotka kuvaavat, miten tietty teksti esitetään esim. lihavoituna. Toiset elementit taas kertovat, minkä tyyppisestä tekstistä on kyse esimerkkinä <h1>Otsikko</h1>. Nämä ovat ns. rakenteellisia merkkauksia. Viimeinen ryhmä on hypertekstit eli linkit. Niitä käyttäen eri dokumentteja voidaan linkittää toisiinsa. HTML-linkkielementti on rakenteeltaan linkin nimi. /9; 16/

2.3 CSS eli kaskadiset tyylitiedostot

CSS, joka on lyhenne englanninkielisistä sanoista cascading style sheet, on tyylikieli, jota käytetään HTML-elementtien ulkonäön määrittämiseen. CSS on keino kertoa selaimelle kuinka HTML-sivu näytetään. Tyylitiedoston ei tarvitse olla erillinen tiedosto vaan se voidaan myös upottaa HTML-koodin sekaan. Upotettaessa CSS-tyylitieto HTML-dokumentin sisään menetetään tyylitiedoston käytön yksi suurimmista eduista eli se, että kun halutaan tehdä muutoksia jonkin sivuston ulkonäköön, muutoksia tarvitsee tehdä ainoastaan yhteen tiedostoon. /12/

2.4 JQuery

JQuery on ilmainen avoimen lähdekoodin kirjasto JavaScript-funktioita, jotka helpottavat monimutkaistenkin asioiden toteuttamista internet -sovelluksessa. JQuery-paketin, joka koostuu yhdestä ainoasta JavaScript-tiedostosta, voi käydä lataamassa osoitteesta jQuery.com. Vaihtoehtoisesti voi käyttää Googlen tai Microsoftin verkossa ylläpitämiä JQuery-luokkia.

JQuery on melko uusi mutta siitä huolimatta hyvin suosittu JavaScript-kirjasto. Alunperin John Resigin vuonna 2006 julkaisema Jquery on tällä hetkellä maailman käytetyin JavaScript-kirjasto. Sen parhaimmat valtit ovat sen alustariippumattomuus ja helppokäyttöisyys. JQuery-kirjaston voi ottaa käyttöön omaan sovellukseen lataamalla kirjaston, joka on vain n. 30 kB pakattuna ja ottaa sen käyttöön kuvan 3 mukaisesti.

```
<head>
<script type="text/javascript" src="jquery.js"></script>
</head>
```

Kuva 3. Paikallisen JQuery-kirjaston käyttöönotto

Vaihtoehtoisesti JQuery-kirjaston voi ottaa käyttöön suoraan joko Microsoftin tai Googlen verkossa ylläpitämien kirjastojen kautta kuvan 4 mukaisesti. Kummassakin tapauksessa käyttöönotossa ei tarvita muuta.

Google

```
<head>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs
/jquery/1.4.2/jquery.min.js"></script>
</head>
```

Microsoft

```
<head>
<script type="text/javascript" src="http://ajax.microsoft.com/ajax/jquery
/jquery-1.4.2.min.js"></script>
</head>
```

Kuva 4. Verkossa ylläpidetyn JQuery -kirjaston käyttöönotto

JQuery on käytännössä keino manipuloida ja muokata HTML-dokumenttia ja sen tyyliä sen jälkeen, kun selain on jo ladannut sivun. JQuery-kirjastosta löytyy keinot helpompaan DOM-elementtien valitsemiseen, animaatioiden lisäämiseen sivulle, tapahtumien käsittelyyn ja Ajax-sovellusten tekoon. /18/

2.5 PostgreSQL

PostgreSQL on ilmainen avoimen lähdekoodin relaatiotietokannanohjausjärjestelmä. PostgreSQL voidaan asentaa kaikille yleisimmille käyttöjärjestelmäalustoille. PostgreSQL on yhdessä Oraclen, MySql:n, SQL Serverin ja DB2 kanssa viisi suosituinta tietokantaa. Järjestys viiden suosituimman kesken vaihtelee hieman lähteestä riippuen.

PostgreSQL:n historia alkaa siitä, kun vuonna 1986 Michael Stonebraker tiimeineen aloitti jatkokehittämään Berkeleyn yliopistossa Kaliforniassa aikaisemmin tutkittua ja kehitettyä Ingres-tietokantaa. Tästä tulee nimi Postgres (Ingresin jälkeen). Ingres oli ollut projekti, jossa oli tarkoitus tuoda uusia uraauurtavia ideoita

kuten oliopohjaista ajattelua tietokantoihin. Tiimi kehitti PostgresSQL:ä kahdeksan vuotta tuoden siihen suuren määrän uusia ominaisuuksia.

Vuonna 1995 kaksi Berkeleyn opiskelijaa, Andrew Yu ja Jolly Chen, vaihtoivat vanhan Ingressin QUEL -pohjaisen tulkin uuteen SQL-tulkkiin. Näin syntyi Postgres95.

Vuonna 1996 Postgres95:n kehitys siirtyi akateemisesta maailmasta avoimen lähdekoodin projektiksi. Joukko kehittäjiä oli huomannut Postgres:n potentiaalin ja lähtivät kehittämään sitä edelleen. Työn tuloksena syntyi PostgreSQL, joka tunnetaan luotettavana ja ominaisuuksiltaan runsaana tietokantana. PostgresSQL:n suosion kasvusta huolimatta MySQL on edelleen kuitenkin suosituin avoimen lähdekoodin tietokanta helppokäyttöisempänä ja nopeampana tietokantana. /15/

PostgreSQL on vähemmän tunnettu ja käytetty tietokanta kuin esimerkiksi MySQL tai Oracle. PostqreSQL:ä pidetään yleisesti hieman hitaampana ja hankalakäyttöisempänä kuin esimerkiksi edellä mainittuja tietokantoja. Se on kuitenkin laajalti käytössä runsaiden ominaisuuksiensa ja luotettavuutensa sekä muokkautuvuutensa ansiosta. PostgreSQL tunnetaan standardeihin vahvasti perustuvana tietokantana. Tällä hetkellä se kattaa suuremman osan ANSI-SQL:2008 standardista kuin sen pahin kilpailija MySQL. /13/

PostgreSQL:n suosituimpia ominaisuuksia ovat mm. MVCC (Multi-Version Concurrency Control). MVCC tarkoittaa sitä, että tietokannan tietueita päivitettäessä vanhaa tietoa ei suoraan ylikirjoiteta uudella tiedolla vaan vanha tieto merkataan suorituksen ajaksi vanhaksi tiedoksi ja se on saatavilla lukuoperaatiolla, jolloin erillisiä luku lukituksia ei tarvita. Uusi tieto taas pysyy uutena versiona, kunnes koko operaatio on valmis ja muutetaan vasta muiden operaatioiden päätyttyä nykyiseksi versioksi. Näin ollen tietokannan eheys säilyy ja tiedot ovat kokoajan käytettävissä lukua varten. Toinen tärkeä ominaisuus on ’point in time recovery’ eli tietokannan voi palauttaa aikaisempana ajankohtana olleeseen tilaan. PostgreSQL:ssä on täysi tuki viiteavaimille (Foreign Key), liitoksille, näkymille, laukaisimille, tallennetuille proseduureille, alikyselyille ja kansainvälisille merkis-

töille. Lisäksi, koska PostgreSQL on avoimen lähdekoodin tuote, siitä löytyy valtava määrä dokumentaatiota ja sitä voi muokata halutessaan omiin tarpeisiin vastaavaksi. /13/

2.6 PDO

PDO eli PHP Data Objects on PHP:n sisäänrakennettu tietokantojen käyttöön tarkoitettu ohjelmointirajapinta. PDO on ollut osana PHP:tä versiosta 5.1 alkaen. PDO:n suurimmat hyödyt saavutetaan siinä, että sen avulla voidaan käyttää lähes kaikkia eri tietokantoja ilman, että jokaiselle tarvitsisi käyttää omia ajureita. PDO on oliopohjainen ja näin ollen sitä on myös helppo laajentaa omilla metodeilla.

PDO:ssa kyselyn suorittamisessa on neljä vaihetta. Ensiksi kysely valmistellaan prepare-metodilla, jolle annetaan parametreina suoritettava SQL-lauseke. Prepare-metodi sekä optimoi kyselyn siten, että sen suoritus kestää mahdollisimman vähän aikaa että muokkaa kyselyn ehtoarvot siten, että ne eivät voi toimia muina kuin arvoina. Tämän tehtyään funktio palauttaa PDOStatement-olion, johon seuraavaksi voidaan sijoittaa oikeat kyselyn hakuehtojen parametrit bindParam-metodilla. Hakuparametreja ei laiteta suoraan hakulausekkeeseen, jotta SQL-injektiot saadaan estettyä. Toisin sanoen bindParam ajaa saman asian kuin esimerkiksi MySQL-rajapinnan mysql_real_escape_string eli lisää tarvittavat kenoviivat ja lainausmerkit, jotta syötteen mahdolliset erikoismerkit menettävät erikoismerkityksensä. Koska kysely on jo tehty turvalliseksi ja tehokkaaksi prepare-metodilla, kyselyn ja hakuehtojen yhdistäminen on turvallista. Tämän jälkeen voidaan suorittaa itse kysely execute-metodilla. Viimeiseksi tiedot luetaan tulosolioon, joko kaikki tulokset tai rivi kerrallaan. Luettaessa kaikki tiedot kerrallaan käytetään fetchAll-metodia, jolloin kaikki haussa saadut tulokset on helposti saatavilla tulosoliossa tietokannassa olleen kentän nimen perusteella. Luettaessa rivi kerrallaan käytetään fetch-metodia, jolloin tiedot voi sijoittaa haluamiinsa muuttujiin. /20/

3 TIETOTURVALLISUUS

Internet on nykyään vihamielinen ympäristö. Erilaiset hakkerit ja haittakoodit yrittävät jatkuvasti tunkeutua käyttäjien koneisiin ja verkossa olevien sovellusten kautta arkaluontoisiin tietoihin käsiksi. Nykyaikana suunniteltaessa ja toteutettaessa, etenkin verkossa olevia sovelluksia, ei voi olla liian varovainen tietoturvallisuuden suhteen. Tässä luvussa kerrotaan, kuinka tätä työtä tehdessä erilaiset uhat on otettu huomioon ja pyritty estämään niiden toteutuminen.

3.1 Turvallinen tietokannan käsittely

Tietokannan käsittelyssä web-sovelluksissa on ensisijaisen tärkeätä varmistua, ettei käyttäjä tai hakkeri pääse ajamaan omia SQL-lauseita tietokantaan eli tekemään ns. SQL-injektiota. SQL-injektio on nykyään suhteellisen yleinen hyökkäysyritys tietokantaa käyttäviä web-sovelluksia vastaan.

Käytännössä SQL-injektiossa pyritään hyödyntämään huonosti tai virheellisesti suunniteltuja tietokantakyselyjä. Käyttäjä tai hakkeri syöttää HTML-lomakkeelle kyselyjen osia tai pahimmassa tapauksessa jopa koko tietokannan tyhjentävän SQL-lauseen. Tämä on mahdollista, mikäli koodissa HTML-lomakkeelta tuleva tieto liitetään suoraan SQL-kyselyyn esim. SELECT * FROM tietokanta WHERE user = $_GET[”user”] AND password = $_GET[“password”]. Tällaisessa tapauksessa hyökkääjä voi kirjoittaa oman SQL-lauseen lomakkeen käyttäjätunnus- tai salasanakenttään ja saada selville sovellukseen sisälle pääsyyn vaadittavat salasanat ja käyttäjätiedot. Syöttämällä edellä olevaan kyselyyn esimerkiksi jotakin’ OR ’1, saadaan vastaukseksi kaikki tietokannassa olevat käyttäjätunnukset. Samaan tyyliin voidaan tietokantaan suorittaa erilaisia operaatioita lomakedatan kautta tapauksessa, jossa tietokannan tietoja päivitetään tai pahimmassa tapauksessa poistetaan lomakkeelta suoraan tulevan datan perusteella ilman tarkistusta, minkälaista data on.

Hyvä perussääntö on, että kaikki käyttäjän syöttämä data voi olla haitallista ja se tulisi aina tarkistaa ennen käyttöä. Tästä syystä mitään käyttäjän syöttämää dataa

ei tulisi suoraan käyttää SQL-kyselyissä. Yksinkertainen tapa vaikeuttaa SQL-injektion tekoa on rajata tai tarkistaa syötetyn datan enimmäispituutta, jolloin mahdollisella hyökkäyksen yrittäjällä on vaikeampi tehtävä saada ujutettua pitkä SQL-lause oikean datan joukkoon.

Toinen hyvä tapa suojautua on lisätä aina syötettävän datan ympärille heittomerkit. Tämä ei kuitenkaan poista sitä ongelmaa, että itse arvo sisältää erikoismerkkejä kuten %-merkkejä, jotka ovat yleisesti tietokannoissa jokerimerkkejä. Tämä ongelma pystytään ratkaisemaan lukemalla merkkijono läpi ja lisäämällä erikoismerkkien eteen kenoviivan(\), joka kertoo, että kyseessä on esimerkiksi %-merkki eikä tietokannan jokerimerkki.

Tässä työssä SQL-injektioilta suojautumiseen käytettiin PDO-kirjaston esikäsittely- ja sitomisfunktioita. Lisäksi varmistettiin, että annettu syöte on sen mittainen, mitä sen oletetaan olevan. Esikäsittelyvaiheessa SQL-lauseeseen ei kirjoiteta suoraan muuttujaa tai saatua syötettä vaan ne korvataan joko kysymysmerkillä (?) tai merkinnällä :arvo esimerkiksi $stmt = $pdo->prepare("SELECT * FROM taulu WHERE arvo=:arvo"); Kun prepare-funktio on esikäsitellyt lauseen eli poistanut tekstistä erityismerkkien erityismerkityksen lisäämällä niiden eteen kauttaviivan ja optimoinut kyselyn, on aika sitoa oikeat arvot kyselyyn. Tämä tapahtuu binparameter-funktiolla. Funktio korvaa esikäsitellyn kyselylausekkeen :arvo tai ? kohdan annetulla muuttujan arvolla. Tämän jälkeen SQL-lausekkeen suoritus on turvallista. PDO-rajapinnassa kysely suoritetaan execute-funktiolla.

Tässä työssä käytettiin ainoastaan SQL-kyselyitä eli tietokantaan ei lisätty, poistettu tai päivitetty mitään. Tietokannan datan lisäämisen ja päivittämisen hoiti NextCall Pro-sovellus. Lähes kaikissa kyselyissä tarvittiin yhtenä syötteenä aikaväliä, jolta tietoja haettiin. Tämä toteutettiin käyttäen JQuery-kirjaston datepicker-luokkaa. Varmistuaksemme datepickerin tuottaman päivämäärän oikeellisuudesta lisättiin valittu päivämäärä tekstikenttään jäsennettynä sellaiseen muotoon, jossa päivämäärä esiintyy myös tietokannassa. Koska kenttää napsauttamalla aukeaa aina datepickerin näkymä, ei kentän arvoa pääse käyttäjä itse muokkaamaan ja

näin ollen tiedämme, että päivämäärä on juuri siinä muodossa kuin sen tulee olla. Mahdollinen hakkeri voi kuitenkin yrittää tehdä oman lomakkeen, jolta lähettää POST-metodilla oman tyyppisen tietonsa ja siksi koodissa tarkistetaan, että tuleva data tulee oikeasta osoitteesta, jolloin kyseinen mahdollisuus estetään. Muita hakukriteerikenttiä ovat esimerkiksi puhelinnumero, jossa tarkistuksena on, että käytettynä on vain numeroita ja enimmäispituutena 14 merkkiä sekä agentti, jossa tarkistetaan syötteen pituutta. Sähköpostiosoitteen perusteella haettaessa kriteerien tekeminen oli hankalampaa. Syötteestä tarkistetaan, että osoite on muotoa jokin_merkkijono@jotakin.tunniste ja, ettei osoite ole yli 30 merkkiä pitkä. Suurin osa muista hakukriteerivalinnoista oli alasvetovalikoissa, joiden arvoja käyttäjä ei päässyt muuttamaan. Alavetovalikoiden tarkistuksessa riittää tarkistus siitä, että tiedot ovat niitä joita on valittavina. Tämä sen takia, ettei tietoja pysty syöttämään http-otsikkotiedon mukana tai muuten pääse muokkaamaan.

3.2 Kirjautuminen ja käyttäjätietojen tarkistus

Koska työssä käsiteltiin sekä käyttäjien henkilökohtaisia statistiikkoja että ryhmien ja sarjojen kokonaisstatistiikkoja, oli työhön sisällytettävä käyttäjätunnuksen ja salasanan vaativa sisäänkirjautuminen. Koska tietokannassa oli jo valmiiksi olemassa käyttäjille tunnukset ja salasanat, päädyttiin työssä käyttämään samoja tunnuksia myös tähän sovellukseen kirjautumiseen. Next Call Pro:ssa on myös mahdollisuus antaa käyttäjille eri rooleja, joten tällä tavalla saatiin eriteltyä kenellä on oikeus mihinkin näkymään. Esimerkiksi puheluhistorianäkymä näkyy kaikille mutta vain omia puheluita pystyy kuuntelemaan. Admin-ryhmään kuuluvilla eli esimiehillä ja valvomokoordinaattorilla on oikeus kuunnella kaikkien käyttäjien nauhoitettuja puheluita. Itse livescreen-sovellus eli perusnäkymä, jossa näytetään sarjojen sen hetkistä tilannetta, näytetään kaikille käyttäjille. Jokainen käyttäjä voi itse valita, mitä sarjoja hän haluaa näytettävän näytöllä.

3.3 PHP ja tietoturvallisuus

PHP-kielisten sovellusten jatkuvasti yleistyessä myös tarve tuottaa mahdollisimman turvallista PHP-koodia lisääntyy. PHP on alkujaan kehitetty tehokkaaksi ja

yksinkertaiseksi ohjelmointikieleksi, jolla kuka tahansa voi tehdä ja julkaista sovelluksiaan internetissä. Koska PHP:tä käytettiin alkujaan lähinnä saamaan internetsivustoihin vuorovaikutteisuutta, kuten esimerkiksi tekemään vieraskirja internet sivustolle, ei tietoturvallisuudella vielä ollut niin suurta merkitystä. Vaikka joku pääsisi muokkaamaan tuota sivustoa, suurin vahinko olisi vieraskirjan sotkeminen tai toimimattomaksi saattaminen. Nykyään PHP:llä toteutetaan laajoja sovelluksia kuten erilaisia ostoskoreja, sivuille rekisteröitymisiä ja kokonaisia yritysten web-portaaleja. Näin ollen PHP-koodia koskevat samanlaiset tietoturvallisuusuhat kuin millä tahansa muullakin kielellä toteutettuja sovelluksia. PHP:llä ohjelmoitaessa tietoturvallisuudessa täytyy vielä ottaa huomioon muutamia pelkästään PHP:n turvallisuusongelmia, kuten esimerkiksi register globals -globaalien muuttujien käyttö tai lähinnä käyttämättä jättäminen. /1/

3.3.1 Register globals

Register globals on PHP- ympäristön asetus, joka voidaan php.ini-tiedostossa laittaa joko päälle tai pois. Register globals -asetuksen ollessa päällä PHP luo automaattisesti globaaleja muuttujia kaikista sille tulevista parametreista. Register globals -asetus kannattaa laittaa jo PHP:n asennuksen aikana pois päältä, koska se on todennäköisesti yleisin turvallisuusriski PHP-sovelluksissa tai ainakin on ollut ennen PHP:n versiota 4.2. PHP 4.2 versiosta eteenpäin register globals asetus on oletuksena pois käytöstä uusissa asennuksissa. PHP 4.1 kehittäjät kehittivät paremman tavan tiedon vastaanottoon ns. superglobaalit-muuttujat $_GET, $_POST, $_COOKIE, $_SERVER ja $_ENV. Näiden muuttujien avulla tiedot voidaan ottaa talteen yksilöllisiin muuttujiin omista lähteistään. Register globals-mekanismilla jokaisesta vastaanotetusta parametrista tehtiin oma parametrin nimen mukainen muuttuja. Tämä on käytännöllistä ohjelmoijan kannalta mutta tietoturvallisuuden kannalta hyvinkin vaarallista. Ongelmia syntyy heti siinä vaiheessa, jos register globals-mekanismilla otetaan vastaan kaksi samannimistä parametria kahdesta eri lähteestä, esimerkiksi, jos sama tieto tulee sekä GET-pyyntönä että evästeeltä. Tällöin vain toinen arvoista välitetään PHP:lle. php.ini -tiedostossa määritettyjen tietojen perusteella valitaan, kumpi syöte menetelmistä

on merkitty tärkeämmäksi. Tärkeämmäksi merkitty tieto valitaan ja välitetään PHP:lle. Vähemmän tärkeäksi merkitty tieto yksinkertaisesti hävitetään. /1/

Toinen vaarallisempi seikka on se, että register globals -mekanismi luo jokaisesta syötteestä oman muuttujan, joka on aivan samanlainen kuin mikä tahansa muu ohjelmoijan luoma muuttuja. Koska PHP:ssa muuttujia ei ole pakko alustaa, voi vahingossa tai tahallaan alustamatta jäänyt muuttuja aiheuttaa suuren vaaran, jos register globals -asetus on asetettuna päälle. Tällöin hyökkääjä voi minkä tahansa syötteen kautta laittaa haluamansa tiedon, arvon tai koodin alustamatta jääneelle muuttujalle ja näin päästä käsiksi arkaluontoiseen tietoon tai mahdollisesti tehdä vakavaa tuhoa sovellukselle. Tässä työssä ei ole käytetty register globals -mekanismia vaan kaikki käyttäjältä tuleva tieto otetaan talteen yksilöityyn superglobaaliin muuttujaan. Register globals -mekanismi on poistettu käytöstä kokonaan palvelimella jossa työ toteutettiin. /1/

Vaikka register globals on nykyään oletuksena pois päältä uutta PHP:tä asennettaessa, on se silti yhä käytössä monella palvelimella. Se on käytössä palvelimilla, joilla ajetaan vanhoja PHP-sovelluksia tai sovelluksia, jotka pohjautuvat vanhoihin sovelluksiin, lähinnä siitä syystä, että sovelluksia ei ole lähdetty nykyaikaistamaan, koska suuresta määrästä koodia on todella vaikeaa ja työlästä etsiä mitkä muuttujat tulisi korvata suberglobaaleilla muuttujilla. /1/

3.3.2 Syöttötiedon tarkistus

Mahdollisesti tärkein asia verkkosovelluksia suunniteltaessa ja toteutettaessa on käyttäjän syöttämän tiedon tarkistaminen. Sovellukset, erityisesti verkkosovellukset, perustuvat useimmiten käyttäjältä tulevaan syötteeseen. Käyttäjältä tulevaa syötettä on pidettävä aina tietoturvauhkana, sillä se voi sisältää mitä tahansa. Syötetty tieto saattaa olla tietoa kalastelevan tai sovelluksen toimintaa vahingoittavan hakkerin ohjelmakoodia tai tavallisen käyttäjän syöttämä matkan varrella korruptoitunut syöte. Tiedon saapuessa PHP-tulkille on se kiertänyt käyttäjän selaimen, mahdollisesti useamman proxy-palvelimen, palomuurin ja filtteröintityökalujen läpi. Mikä tahansa näistä vaiheista voi joko tahallisesti tai tahattomasti muokata

käyttäjän syöttämää tietoa ja näin ollen vahingoittaa sovellusta. Tästä syystä kaikki koodin ulkopuoliset syötteet tulee tarkistaa mahdollisimman hyvin ja tarkasti. Mitään täysin varmaa tekniikkaa tai tapaa tähän ei ole olemassa. Mitä pitempi ja monipuolisempi annettu syöte on, sitä vaikeampi sen oikeellisuutta on tarkistaa. Tässä mielessä valmiiksi annetut vaihtoehdot, kuten alasvetovalikot ovat helpoimpia sillä niissä voidaan helposti tehdä tarkistus, onko annettu syöte täysin annettuja vaihtoehtoja vastaava. Pitkän vapaan tekstikentän tekstiä taas on huomattavasti vaikeampi tarkistaa. /1/

Paras lähestymistapa tiedon tarkistuksessa ei ole ”blacklistaus” eli ei toivottujen syötteiden poissulkeminen vaan ”whitelistaus” eli sallittujen syötteiden salliminen. Tämä siitä syystä, että kun sallitaan vain varmasti hyväksytyt syötteet, ei läpi pääse sujahtamaan sellaista dataa, joka on haitallista mutta, josta ei vielä tiedetä ja näin ollen sitä ei ole estettyjen listalla. On kuitenkin harmittomampaa, että jokin oikeanlainen syöte jää hyväksymättä kuin se, että haitallinen syöte pääsee sotkemaan sovelluksen toimintaa ja kenties tuhoamaan koko sovelluksen tai yhtiön tärkeitä tietoja sisältävän tietokannan sisällön.

3.3.3 Cross-Site Scripting

Cross-site Scripting (XSS) on yksi yleisimmistä web-sovellusten haavoittuvuuksista. XSS- haavoittuvuus tarkoittaa, sitä että hyökkääjä tai hakkeri pyrkii tallentamaan omaa CSS-, HTML- tai JavaScript-koodia web-sovelluksen käyttämään tietokantaan. Sovelluksen tulostaessa tämän koodin myöhemmin näytölle, esimerkiksi osana vieraskirjaviestiä, se sitten muokkaa sivua tai ajaa jotakin koodia, jolla yleensä pyritään varastamaan käyttäjätietoja tai ohjaamaan käyttäjän jonkin kolmannen osapuolen sivustolle. XSS-haavoittuvuudessa hyökkääjä siis käyttää hyväkseen web-sovellusten tapaa tulostaa näytölle käyttäjän syötteitä ilman sen suurempia tarkistuksia.

XSS-haavoittuvuuksia on kahdentyyppisiä, ns. suora hyökkäys, jossa annettu syöte tulostetaan ainoastaan syöttäneelle käyttäjälle, ja tallennettu hyökkäys, jossa annettu syöte näkyy kaikille tuleville käyttäjille. Suoralla hyökkäyksellä pyritään

yleensä saamaan tietoa sovelluksesta tai verkkosivustosta, jota voidaan myöhemmin käyttää vielä vakavamman hyökkäyksen tekemiseen. Tallennetulla hyökkäyksellä pyritään yleensä varastamaan käyttäjätietoja tai ohjaamaan käyttäjä toiselle vaarallisemmalle sivustolle.

PHP tarjoaa onneksi tehokkaita ja melko yksinkertaisia keinoja torjua XSS-haavoittuvuuksia. Näitä keinoja ovat erilaiset merkistökoodausfunktiot ja funktiot, jotka poistavat tai muokkaavat tekstistä merkkejä, joilla on erityismerkitys HTML:ssä. Näistä funktioista yksi on htmlspesialchars-funktio. Funktio poistaa annetusta syötteestä kaikki HTML:n erikoismerkit (&, <, >, ”, ’) ja korvaa ne HTML-entiteeteillä kuten esimerkiksi & korvataan & -merkillä. Tämän jälkeen merkki on vain tavallista tekstiä eikä enää osa sivun koodia eikä se näin ollen pysty tekemään mitään haitallista.

Jotta pystytään suojautumaan JavaScriptin ja CSS:n syöttämiseltä, pitää huomioida mahdollisuus, että osa annetuista syötteistä päätyy usein HTML-elementtien attribuuttien arvoksi. Tästä hyvänä esimerkkinä mainittakoon tapaus, jossa käyttäjä syöttää vieraskirjaan URL:n, joka osoittaa tiettyyn sivustoon. Usein tällaisessa tapauksessa syötteestä tehdään suoraan <a> elementin href -attribuutin arvo eli linkki. Riskinä on, että käyttäjä syöttää väliin lainausmerkin (”), joka päättää attribuutin, jos href –attribuutti on aloitettu samanlaisella merkillä. Tällöin selain päättää sen attribuutin ja aloittaa uuden. Lainausmerkin jälkeen voi käyttäjän syötteessä olla erilaisia uusia attribuutteja tai JavaScript funktioiden kutsuja. Haavoittuvuus voidaan estää käyttämällä htmlspesialchars-funktiota, joka muuttaa lainausmerkin (”) " -merkkijonoksi, jolloin se ei katkaise lainausta. Käytettäessä href-atribuutissa heittomerkkiä (’), täytyy htmlspesialchars-funktiolle antaa erikseen ENT_QUOTES -parametri, jotta se muokkaa myös heittomerkit. Tästä syystä attribuuttien arvot kannattaa aina merkitä lainausmerkein. /1/

3.3.4 Istuntojen turvallisuus

Istunnot (eng. Sessions) ovat kätevä tapa tunnistaa ja seurata web-sovelluksen käyttäjää. Istuntomekanismilla käyttäjä kyetään pitämään tunnistettuna koko is-

tunnon ajan. Istunnot toimivat siten, että kun käyttäjä kirjautuu sovellukseen erilaista tietoa kirjautumisesta ja hänen tekemisistään sovelluksessa tallennetaan PHP-koodissa superglobaaliin taulukkoon $_SESSION. Tieto pysyy tässä muuttujassa kunnes käyttäjä poistuu sivulta, kirjautuu ulos tai istunto vanhenee. Itse PHP-koodin kannalta istunnot ovat yksinkertaisia käyttää.

Taustalla tapahtuu paljon asioita, joita on hyvä ottaa huomioon $_SESSION taulukkoa käytettäessä. Yleisimmin istunnon tiedot tallennetaan selaimen evästeisiin (eng. Cookie) sessiota luotaessa. Tämä tapahtuu siten, että sessiota luotaessa palvelin lähettää Set-Cookie -otsaketiedon muiden otsikkotietojen mukana, jolla pyydetään käyttäjän selainta luomaan eväste. Jos selain hyväksyy pyynnön, se kirjoittaa evästetiedon omaan evästetietovarastoon, joka yleensä on tavallinen tekstitiedosto. Tiedosto on täysin ihmisen ymmärtämää tekstiä, joten helpoin tapa varastaa toisen käyttäjän evästetiedot on käydä lukemassa tiedot käyttäjän tietokoneelta. Ongelma on lähinnä yhteiskäytössä olevilla tietokoneilla kuten esimerkiksi koulujen tai kirjastojen koneilla. Tällainen istunnon varastaminen on helpoiten estettävissä laittamalla evästeen elinikä mahdollisimman lyhyeksi tai menneisyyteen, jolloin eväste tuhotaan heti, kun käyttäjä poistuu sivustolta tai sulkee selaimen. Eväste tuhotaan myös, kun sessio tuhotaan, joka kannattaa tehdä koodissa uloskirjautumisen yhteydessä.

Evästetietoja pyritään myös kalastelemaan palvelimen ja asiakassovelluksen välisestä liikenteestä. Käytettäessä HTTP-protokollaa tieto kulkee täysin salaamattomana, jolloin erilaisilla tietoliikennepakettien sieppausohjelmilla ja viruksilla voidaan varastaa käyttäjän eväste. Tämä voidaan estää käyttämällä salattua https-protokollaa, jolloin lähetettävä tieto salataan käyttäen joko SSL- tai TLS-salausta. /1; 17/

3.3.5 Code Injection

”Code injection” on PHP-sovellusten vaarallisin haavoittuvuus, koska hakkeri pyrkii syöttämään omaa koodiansa sovellukseen. ”Code injection” haavoittuvuudessa käytetään hyväksi PHP:n include-, require- ja eval-funktioita, joilla eri tie-

dostoissa sijaitsevat koodit liitetään yhteen. Sovelluksessa jossa käytetään ”register globals”-mekanismiä ja tiedostoja liitetään sovellukseen muuttujan arvon perusteella, on ”Code injectionin” teko erittäin helppoa. Hakkerin tarvitsee vain syöttää samannimisenä POST- tai GET-parametrina haittakoodin osoite, kuin millä sovelluksessa liitetään tiedostoja sovellukseen include-, require- tai eval-funktioilla. ”Code injection:n” estämiseksi, tiedostoja ei kannata sovellukseen liittää suoraan muuttujasta, jos syötettä ei ensin tarkisteta. Parempi tapa valita liitettävä tiedosto on tarkistaa muuttujan arvo ja sen perusteella valita mikä ennalta määritetty tiedosto sovellukseen liitetään. ”Code injection” haavoittuvuus ei ole pelkästään ”register globals” -mekanismiä käyttävien sovellusten ongelma. Haavoittuvuuden käyttäminen vain on vaikeampaa, jos ”register globals” on poissa käytöstä. Parhaita tapoja estää ”Code injection” on käyttää aina koko saantipolkua, kun sovellukseen liitetään muita tiedostoja, sekä välttää liittämästä tiedostoja suoraan muuttujan arvolla tai ainakin tarkistaa ensin, että muuttujan arvo on oikeanlainen. /1/

4 TYÖN SUUNNITTELU

4.1 Suunnittelun lähtökohta

Työn suunnittelun ja toteutuksen osalta annettiin melko vapaat kädet. Työn lähtökohtana oli saada kahdelle suurelle näytölle Anvian valvomoon näytöt, joissa voidaan näyttää reaaliaikaista tietoa eri sarjojen tilanteesta. Näkymän tulisi olla yksinkertainen ja siitä tulisi selvitä kuvan 5 mukaiset kentät. Kuvassa 5 näkyviä kenttiä lisättiin myöhemmin, jotta kaikki tarpeellinen tieto saadaan näkyviin. Sarjoja tulee jatkuvasti lisää ja vanhoja saattaa poistua, joten sarjojen lisääminen ja poistaminen sovelluksesta pitää onnistua helposti. Tarkoitus ei ole näyttää kaikkien Anvialla käytössä olevien sarjojen tilannetta kerralla, joten tarvitaan myös näytettävien sarjojen valitsemisen mahdollisuus. Lisänä tässä työssä on ollut tarkoitus tehdä erilaisia raportointiin ja historiatietojen seurantaan liittyviä hakutoimintoja Next Call Pro:n PostgreSQL-kannasta. Nämä toteutetaan käyttäen osittain samoja luokkia ja metodeja kuin itse livescreen-sovelluksessa.

Esim. Oma NCP LiveScreen

	PUHELUITA JONOSSA	Jonotus aika	VASTAUS % (vuorokausi 00-23.59)	Puhelu määrä (vuorokausi 00-23.59)
VIKA (10019)	5 kpl	2:00 min	81%	121
Helpdesk (FIN/SWE)	2 kpl	2:58 min	81%	60
On Cable	0 kpl	1:30 min	81%	12
[redacted]	1 kpl	2:00 min	81%	12

Kuva 5. Livesreen-sovelluksen esimerkki, jonka pohjalta työtä lähdettiin toteuttamaan.

Työn ensimmäisessä vaiheessa suunniteltiin itse livescreen-sovellus ja siihen sisäänkirjautuminen. Kirjautuminen tehdään NextCall Pro:n postgres -tietokantaa vasten. Tietokannasta löytyi valmiina käyttäjätunnukset ja salasanat, joita käytetään itse pääsovelluksessa, joten niitä päädyttiin käyttämään myös tähän sovellukseen kirjautumisessa. Koska kirjautumiseen käytetään samoja tunnuksia kuin NextCall Pro:hon kirjautuessa, ei niitä tarvitse hallinnoida erikseen vaan tunnusten luonti ja muokkaus tapahtuu kumpaankin sovellukseen samalla. Myös käyttäjän kannalta on helpompaa, koska ei tarvitse muistaa useita eri tunnuksia.

Kirjautuminen toteutettiin HTTP Authentication -menetelmällä. Menetelmä todettiin riittävän turvalliseksi, koska sovellus tulee näkymään ainoastaan Anvian sisäverkossa. Menetelmässä PHP:n heder-funktiota käytetään kuvan 6 esittämällä tavalla lähettämään selaimelle tieto, että tämä sivusto vaatii kirjautumista. Kuvan esimerkkikoodi aiheuttaa sen, että selain näyttää käyttäjälle ikkunan, jossa pyydetään käyttäjätunnusta ja salasanaa. Kun käyttäjä on syöttänyt tunnuksen ja salasanan ja painanut kirjaudu-painiketta, selain kutsuu uudestaan PHP:skriptiä, josta pyyntö tuli. Selain asettaa syötetyt arvot esiasetettuihin muuttujiin PHP_AUTH_USER, PHP_AUTH_PW ja AUTH_TYPE. Tämän jälkeen näitä arvoja voidaan käyttää PHP:koodissa globaalista taulukosta $_SERVER.

```
header('WWW-Authenticate: Basic realm="'.$auth_realm.'"');
header('HTTP/1.0 401 Unauthorized');
```

Kuva 6. HTTP Basic Autchentication.

Itse kirjautumiskoodi on tehty siten, että aluksi luodaan uusi istunto eli sessio, jolloin syntyy globaali taulukko $_SESSION, johon voidaan tallentaa tietoa, joka pysyy tallessa niin kauan kunnes siirrytään pois tästä sovelluksesta tai sessio tuhotaan. Kun sessio on luotu, lähetetään edellä mainitulla tavalla HTTP Authentication -pyyntö selaimelle. Tämän jälkeen tarkistetaan, että käyttäjä on syöttänyt arvot

kumpaankin kenttään. Jos kummassakin muuttujassa on sisältöä, tehdään seuraavaksi muutamia tarkistuksia syötteiden oikeellisuudesta sekä tarkistetaan, etteivät ne sisällä mitään haitallisia merkkejä. Tämän jälkeen suoritetaan kysely tietokantaan. Mikäli annettu käyttäjätunnus ja salasana täsmäävät tietokannassa oleviin arvoihin asetetaan käyttäjätunnus $_SESSION ['kayt_tunnus'] kenttään, muutoin annetaan virheilmoitus ja kysytään tunnistustietoja uudestaan. Kun käyttäjätunnus on asetettu session muuttujaan, sitä voidaan käyttää sovelluksessa sivujen uudelleen latauksen jälkeen, kunnes sessio tuhotaan uloskirjautumisen yhteydessä. /16/

4.2 Käytettävien menetelmien valinta

Käytettävien menetelmien suunnittelussa oli alusta asti selvää, että sovellus tullaan toteuttamaan PHP-kielellä ja tietokannan valintaa ei tarvinnut suunnitella, sillä kaikki tieto oli Next Call Pron PostgreSQL-tietokannassa. Tietokantahakujen luomiseen ja testaamiseen otettiin käyttöön pgAdmin 3 sovellus, joka on tarkoitettu PostgreSQL-tietokantojen ylläpitoon ja tietokantahakujen testaamiseen ja luontiin. Tarkoitus oli ensin varmistaa, että kaikki tarvittava tieto löytyy tietokannasta ja saada luotua tietokantakyselyt, joilla tieto saadaan haettua mahdollisimman lyhyessä ajassa. Usean testauksen ja tietokannan tutkimisen jälkeen suorituskykyiset ja oikean tuloksen tuottavat kyselyt löytyivät. Tietokantayhteyksien luomiseen ja kyselyjen suorittamiseen tietokannan ja PHP-koodin välillä otettiin käyttöön PDO-driver -luokka, joka on yleisesti käytössä Anvialla. Tärkein syy PDO-driverin valintaan oli sen turvallisuusominaisuudet, joista kerrotaan tarkemmin kappaleessa 2.7 ja 3.1. Sovelluksen ulkoasun toteutukseen ja hallintaan otettiin käyttöön CSS-tyylitiedosto, jotta ulkoasua voidaan hallita ja muokata yhdestä tiedostosta.

5 SQL-KYSELYIDEN LUONTI JA NIIDEN TEHOSTAMINEN

Päänäkymän ja kirjautumisen suunnittelun ja toteutuksen jälkeen tässä työssä lähdettiin tutkimaan tietokantaa ja etsimään sieltä tarvittavia tietoja ja luomaan SQL-kyselyjä, joilla tiedot tietokannasta saadaan haettua. Tässä käytettiin apuna pgAdmin 3 –sovellusta, jolla pystytään tutkimaan tietokantaa graafisen käyttöliittymän kautta sekä luomaan ja ajamaan SQL-kyselyjä. Koska työssä käytetty tietokanta on käytössä myös NextCall Pro -sovelluksessa ja siksi jo valmiiksi suuren rasituksen alla, on tärkeää, että SQL-kyselyt eivät kuormita tietokantaa liikaa. Tästä syystä sovelluksessa pyritään välttämään sekä SQL-funktioiden että sisäisten kyselyiden käyttöä ja pitämään tietojen päivitysväli mahdollisimman suurena. On huomioitava, ettei päivitysväli saa olla liian suuri, jottei sovelluksen idea reaaliaikaisen tiedon näyttäjänä kärsi. Tietokannan kannalta on tässä tapauksessa parempi, että kyselyitä tehdään useita peräkkäin kuin, että tehtäisiin yksi raskas ja pitkään kestävä kysely.

Funktioiden, kuten esimerkiksi count-funktion, käyttö SQL-kyselyissä lisää huomattavasti kyselyn suorittamiseen vaadittavaa tietokannan kuormitusta. Kuormitus lisääntyy etenkin, mikäli funktio joutuu käsittelemään suuren määrän tietueita. Count-funktio hakee kaikki hakukriteerit täyttävät tietueet tietokannasta ja laskee montako niitä löytyi. Jotta tietokannan kuormitus pysyisi mahdollisimman pienenä, päädyttiin kaikki mahdollinen tiedon analysointi ja laskeminen tekemään itse PHP-koodissa.

5.1 Kyselyiden tehostamisen menetelmät

Tietokantakyselyjä luodessa kannattaa merkitsevimmät hakuehdot laittaa ensimmäiseksi. Tällöin kyselyoptimoija voi jo heti aluksi poistaa tuloksesta mahdollisimman paljon tietueita ja näin haun välitulos on heti aluksi mahdollisimman pieni. Välituloksen ollessa pieni sen läpikäyminen seuraavan ehdon kohdalla on huomattavasti nopeampaa, kuin suuren välituloksen läpikäynti.

Tietokantakyselyn tuloksena tulee aina palauttaa vain välttämättömät kentät. Kyselyissä ei siis koskaan tule käyttää muotoa *SELECT * FROM kayttajat*, vaan valita vain ne kentät, joiden tietoa tarvitaan. Esimerkiksi *SELECT nimi, osoite, kunta FROM kayttajat*. Mitä enemmän kenttiä tietokanta joutuu käsittelemään kenttälistan takia, sitä enemmän töitä se joutuu tekemään toimenpiteen suorittamiseksi.

5.2 Taulujen indeksit ja liitokset

Relaatiotietokannassa tiedot lähes poikkeuksetta sijaitsevat eri tauluissa. Yleensä yhden taulun tiedot liittyvät aina muihin tauluihin. Taulujen väliset liitokset tehdään avaimien avulla. Lapsitaulussa oleva viiteavain viittaa isätaulun perusavaimeen. Avaimet ovat aina yksilöllisiä yleensä numeroarvoja, joilla tiedot voidaan yhdistää. Nykyään useissa tietokantatuotteissa avaimille luodaan automaattisesti indeksit eli hakemisto, jonka avulla tietokannan taulun käsittelyä nopeutetaan. Indeksejä voidaan luoda muillekin sarakkeille kuin avain sarakkeille, esim. jos tietoa usein haetaan sukunimen ja etunimen perusteella, voidaan luoda indeksi näille sarakkeille yhdessä ja näin nopeuttaa hakuja.

Useasta eri tietokantataulusta tietoa haettaessa tulee aina käyttää taulun avainta liitoksen luomiseen muihin tauluihin, jotta kysely voidaan suorittaa mahdollisimman tehokkaasti ja tulokseksi saadaan juuri haluttu tieto. Koska avain on yksilöllinen numero, joka vastaa toisessa taulussa olevaa kentän arvoa saadaan valittua juuri haluttu rivi taulusta. Liitokset tehdään SQL-kyselyn WHERE-osiossa. Kuvan 7 mukaisessa tapauksessa *SELECT ag.nimi, c.tapahtuma, c.aloitus_aika FROM agentti ag JOIN cases c ON* ***ag.agentti_id = c.agentti_id****;* Kysely palauttaa tiedot tapahtuman hoitaneesta agentin nimestä, tapahtuman aloitusajan ja tapahtuman tyypin.

Agentti-taulu

agentti_id	nimi	alanumero
1	Jaakko	2345
2	Ville	4353

Cases-taulu

case_id	agentti_id	tapahtuma	aloitus_aika	sarja_id
1	1	puhelu	07-07-2011 14:03:00	1234
2	1	s_posti	07-08-2011 12:25:05	2534
3	2	puhelu	07-09-2011 14:06:38	1234

Kuva 7. Yksinkertaistettu esimerkki taulujen liitoksista.

Taulujen indeksien luontiin ja muokkaamiseen tässä työssä ei ollut mahdollisuutta vaikuttaa, koska käytössä on NextCall Pro:n valmis tietokanta, johon meillä ei ole mahdollisuutta tehdä muutoksia. Tietokantakyselyt kannatta pitää mahdollisimman yksinkertaisena ja mieluummin tehdä useita pieniä kyselyitä kuin yksi suuri ja raskas kysely. Pienet kyselyt on sekä ohjelmoijan kannalta helpompi ylläpitää ja muokata että tietokannan kannalta parempia, kuin suuret monimutkaiset kyselyt. Ohjelmoija pystyy pienemmissä kyselyissä paremmin hahmottamaan kyselyn kokonaisuuden ja näin käyttämään paremmin hyödyksi indeksejä. /6/

5.3 SQL-funktioden käyttö ja datatyyppimuunnokset

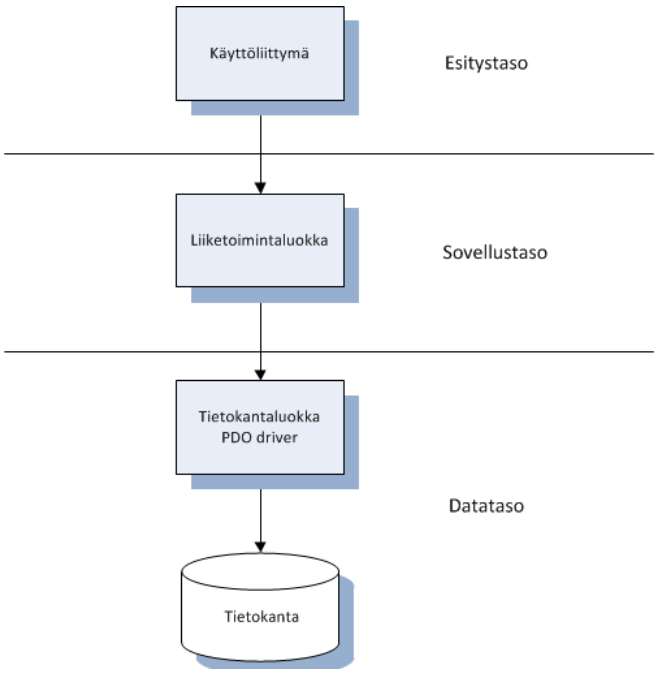
SQL-kyselyissä on mahdollista käyttää erilaisia aritmeettisia operaatioita eli erilaisia lasku- ja muunnostehtäviä. Näihin operaatioihin kuluu aina aikaa ja tietokannan resursseja, etenkin, jos operaatio tehdään suurelle joukolle dataa. Niinpä tässä työssä kaikki tiedon vertailu ja muokkaus sekä tyyppimuunnokset tehdään PHP-koodissa eikä tietokantakyselyiden yhteydessä tietokannassa. Kyselyissä pyritään myös mahdollisuuksien mukaan käyttämään ehtoina vain saman tyyppistä dataa, jotta vältytään tyyppimuunnoksilta, jotka myös syövät tietokannan suorituskykyä.

5.4 Optimointityökalu

Tietokantakyselyiden tehostamisessa käytettiin tässä työssä apuna Postgre SQL:n optimointityökalua. Optimointityökalu toimii siten, että kyselyä suoritettaessa kyselyn eteen kirjoitetaan EXPLAIN, jolloin optimoija käy kyselyn läpi ja kertoo kuinka se suoritetaan ja kauanko suoritukseen kuluu aikaa. Arviota tehokkaimmasta suoritustavasta kutsutaan suoritussuunnitelmaksi. Suoritussuunnitelma koostuu useista yksittäisten tietokantaoperaatioiden saantipoluista. EXPLAIN-komennolla tietokanta kertoo tehokkaimmaksi arvioimansa tavan suorittaa kyselyn, jokaisen operaation suorittamiseen kuluneen ajan, suorituksessa käytettyjen levyoperaatioiden määrään sekä hyödynnetyt vertailuehdot. Optimointityökalun tulosten perusteella löydettiin tässä työssä käytetyistä SQL-kyselyistä tehokkaimmat. /14/

6 SOVELLUKSEN TOTEUTUS

Sovellus toteutettiin kolmitasomallin (eng. three tier) mukaan. Kolmitasomallissa käyttöliittymä, liiketoimintakerros (bisneslogiikka) ja tietokantayhteys-metodit jaetaan kaikki omiin luokkiinsa. Käyttöliittymäluokka hoitaa tiedon näyttämisen, jonka se saa liiketoimintaluokalta. Käyttöliittymäluokka ei ole koskaan suoraan yhteydessä tietokantaan tai tietokantaluokkaan vaan kaikki kommunikointi tapahtuu liiketoimintaluokan kautta. Kolmitasomallin keskeisin idea on siinä, että tiettyä osaa sovelluksesta voidaan muuttaa tai vaihtaa kokonaan ilman, että muutoksia tarvitsee tehdä kaikkiin luokkiin. Esimerkiksi tietokantaa voidaan näin ollen helposti vaihtaa ilman, että käyttöliittymä- tai liiketoimintaluokkiin tarvitsee tehdä muutoksia. Samoin itse sovelluksen toteuttaminen ja suunnittelu on helpompaa sovelluksen ollessa jaettuna selkeisiin osiin. Tämän työn osalta kolmitasomallin käyttö oli onnistunut ratkaisu, sillä näin voitiin toteuttaa ensin käyttöliittymä- ja liiketoimintaluokat ja erillisenä kokonaisuutena tietokantaluokka ja tietokantakyselyt. Lopuksi nämä yhdistettiin ilman suurempia ongelmia. Kolmitasomallin käyttö mahdollisti myös sen, että sovellusta pystyttiin testaamaan ja esittelemään vaikka se ei kaikilta osin vielä toiminutkaan.



Kuva 8. Kolmitasomalli

6.1 Käyttöliittymäluokat

Sovelluksessa on kaksi käyttöliittymäluokkaa. Näyttöä varten omansa ja raportointityökalua varten omansa. Käyttöliittymäluokat ovat osa samaa sovellusta ja ne käyttävät samaa liiketoimintaluokkaa, joka taas käyttää yhteistä tietokantaluokkaa. Liiketoimintaluokassa on funktiota, joita kumpikin käyttöliittymä käyttää yhteisesti sekä funktioita, jotka ovat vain toisen käytössä. Käyttöliittymät käyttävät myös samaa sisäänkirjautumisluokkaa, joka taas käyttää samoja liiketoiminta- ja tietokantaluokkia kuin käyttöliittymät itse.

Sovellusta käynnistettäessä käyttäjältä kysytään ensimmäisenä käyttäjätunnusta ja salasanaa. Tunnukset ovat samat kuin NextCall Pro –sovelluksessa, joten niitä ei tarvitse erikseen ylläpitää tai luoda tätä sovellusta varten. Käyttäjän kirjautuessa ensimmäistä kertaa sovellukseen hänelle aukeaa ensimmäisenä näytettävien sarjojen valintaikkuna. Toisin sanoen, jos sovellukselle luodusta MySQL-tietokannasta ei löydy kyseiselle käyttäjälle merkintää, pyydetään häntä valitsemaan haluaman-

sa sarjat. Käyttäjä pystyy valitsemaan vain niitä sarjoja, joihin hänellä on oikeus kirjautua NextCall Pro:ssa. Näytettävien sarjojen valintaikkunan toteutuksesta kerrotaan enemmän tämän kappaleen lopussa. Käyttäjä, joka on kirjautunut jo aikaisemmin ja valinnut sarjat, ohjataan suoraan livescreen-näyttöikkunaan ja hänelle näytetään kuvan 9 mukainen näkymä niistä sarjoista, jotka hänen tunnuksensa perusteella on tietokannassa tallennettuna. Näitä tietoja voi muokata näytettävien sarjojen valintaikkunan kautta.

	Puheluita Jonossa	Jonotusaika	Vastaus % 60s (vuorokausi 00-23:59)	Puhelu määrä (vuorokausi 00-23:59)	Vastaajia sarjassa
VIKA (10019)	0	0 min	79.75 %	158	0
Helpdesk (FIN/SWE)	0	0 min	70.83 %	48	0
HD On Cable	0	0 min	100.00 %	5	0
HD Sijaintitietopalvelu	0	0 min	88.89 %	18	0
[redacted]	0	0 min	100.00 %	12	0
HD Sisäinen sarja	0	0 min	88.89 %	9	0
[redacted]	0	0 min	100.00 %	2	0
[redacted]	0	0 min	0.00 %	1	0
Yhteensä/keskimäärin	0	0.00 min	78.54 %	253	0

Kuva 9. Livescreen päänäkymä.

Next Call Pro:ta käyttää Anvialla usea eri organisaatio. Jokaisella organisaatiolla on useita omia sarjoja joten, jotta sovellusta voidaan käyttää eri organisaatioissa, kuten valvomossa, helpdeskissä, asiakaspalvelussa ja laskutuksessa, täytyy sovelluksessa olla mahdollisuus valita mitä sarjoja näytetään. Sarjojen näytön valinnan olisi voinut toteuttaa myös pelkästään sen perusteella, mistä organisaatiosta kirjautunut käyttäjä on. Koska sarjoja on joka tapauksessa tarve kyetä lisäämään ja poistamaan, päädyttiin sarjojen valinta tekemään käyttäjäkohtaiseksi.

Näytettäviä sarjoja valittaessa on sovelluksessa ensimmäiseksi haettava NextCall Pro:n tietokannasta sarjojen nimet ja niitä vastaavat id:t. Sarjoja löytyy tietokannasta niin paljon, että on järkevää näyttää vain kirjautuneen käyttäjän oikeuksia vastaavat sarjat valinnassa eli vain ne sarjat mihin hän pystyy kirjautumaan NextCall Pro:ssakin. Nimet näytetään valintanäkymässä, jossa niitä voi valita valintaruutujen avulla. Oletuksena on valittuna sillä hetkellä kirjautuneelle käyttäjälle valitut sarjat. Tämä tieto haetaan erillisestä MySQL–tieokannasta, jossa on kunkin käyttäjän perusteella tieto siitä, mitä sarjoja hän on valinnut aikaisemmin. Sarjojen id:t ovat lomakkeella olevien checkbox-elementtien value -atribuuttien arvoina, jolloin ne saadaan otettua talteen, kun käyttäjä valitsee haluamansa sarjat ja lähettää tiedot painamalla tallenna-painiketta. Valittujen sarjojen id:t ja nimet tallennetaan MySQL-tietokantaan, josta livescreen-näytössä voidaan hakea tämä tieto ja näyttää valitut sarjat. Tietoa ei voida tallentaa NexCall Pro:n omaan tietokantaan, koska toteutetulla sovelluksella ei ole oikeutta kirjoittaa sinne tietoa ainoastaan lukea. Siksi otettiin käyttöön erillinen tietokanta, jonne tallennetaan jokaisen käyttäjän perusteella sarjat ja niiden id:t, jotka hän on valinnut.

6.1.1 Livescreen-sovelluksen käyttöliittymä

Livescreen-sovellusta työstäessä lähdettiin toteuttamaan livescreen-sovelluksen käyttöliittymäluokkaa eli sovelluksen ulkonäköä. Sovelluksen päänäkymän eli livescreen -näytön runko toteutettiin kolmella html div -elementillä. Sovelluksen yläreunassa on div-elementti, jossa näytetään linkki näytettävien sarjojen valintaikkunaan, kirjautunut käyttäjä sekä uloskirjauspainike. Keskellä eli yläpalkin alapuolella on toinen div-elementti, jossa näytetään käyttäjän valitsemien sarjojen tilastotietoja reaaliajassa kuvan 9 mukaisesti. Tiedot päivittyvät minuutin välein. Alareunassa on vielä yksi div-elementti, jossa kerrotaan kauanko sivun lataus ja tietojen haku kesti. Käyttöliittymäluokkaa toteutettaessa ja testattaessa luotiin samalla myös runko liiketoimintaluokalle ja sen toiminnalle. Tietoja ei kuitenkaan tässä vaiheessa vielä haettu tietokannasta. Liiketoimintaluokan set-metodit tuottivat ainoastaan oikeaa vastaavaa tietoa. Tiedot olivat todellisuudessa vain PHP:n rand-funktion tuottamia arvoja. Rand-funktiota käytettiin siksi, että tiedot saatiin

vaihtumaan ja näin ollen saatiin testattua hälytysten toiminta arvojen ollessa liian suuria tai pieniä. Näillä menetelmillä saatiin sovelluksen runko toimimaan niin hyvin, että vaihdettaessa oikeaan tietokannasta haettuun tietoon tarvitsi tehdä muutoksia vain liiketoimintaluokkaan sekä toteuttamaan itse tietokantaluokka. Tältä osin kolmitasomallin idea toteutui mainiosti.

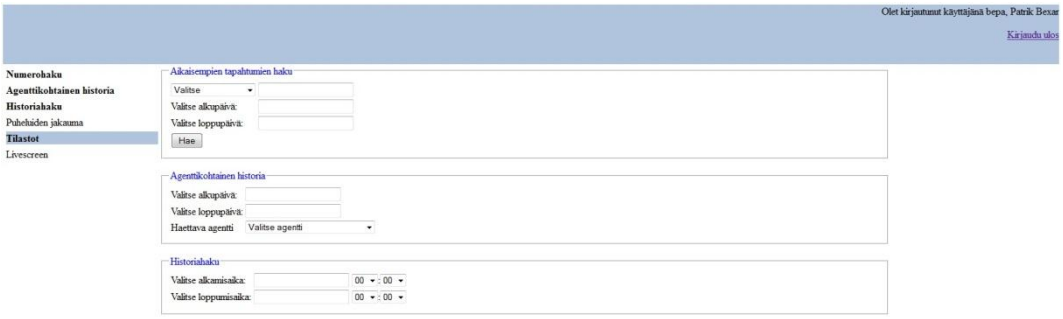
Tietoja ei haettu suoraan tietokannasta osittain siitä syystä, että käyttöliittymäluokka haluttiin toteuttaa ensin ja hyväksyttää sovelluksen tilaajalla. Mutta osittain myös siksi, koska tietokantaan ei saatu vielä yhteyttä sovelluksesta. Anvian sisäinen palomuuri osoittautui syyksi siihen, että TCP/IP-liikenne ei kulkenut palvelinten välillä. IP-protokollan päällä ajettavaa TCP-protokollaa käytetään tiedon siirtoon palvelimien välillä, tässä tapauksessa tiedon hakemiseen NextCall Pro-palvelimelta palvelimelle, jossa toteutettu sovellus sijaitsee. Kun palomuurista avattiin reitti palvelimelle ja lisättiin staattinen reitti NextCall Pro:n ja käytetyn palvelimen välille saatiin yhteys toimimaan. Palomuurin avaukset ja reitin määritykset tekivät Anvian tietohallinnossa näitä asioita hoitavat henkilöt.

6.1.2 Raportointityökalun käyttöliittymä

Raportointityökalun käyttöliittymän perusta on samoin kuin livescreen-sovelluksen perusta rakennettu kolmen div-elementin pohjalle. Sovelluksen yläreunan palkki on oma div-elementtinsä, jossa näytetään, kuten livesreen-sovelluksessa, kirjautunut käyttäjä ja uloskirjautumispainike. Vasemmassa reunassa oleva div-elementti taas sisältää sovelluksen valikkorakenteen. Loput näytön alueesta on varattu sisältö div-elementille, jossa näytetään käyttäjän valintojen mukaan käytettävissä olevat hakuvaihtoehdot ja niiden antamat tulokset.

Valikko toteutettiin p-elementeillä, joita muokataan tyylitiedoston ja jQuery-funktioiden avulla. Kaikilla p-elementeillä on sama class-attribuutin arvo ”menu”, jolla ne tunnistetaan valikon elementeiksi. Hover- ja addClass-funktioiden avulla menu-luokan p-elementit saadaan korostumaan ja taustaväriksi vaihtuu sininen, kun käyttäjä vie hiiren osoittimen elementin päälle. Käyttäjän painaessa hiiren painiketta elementin päällä, näytetään tai piilotetaan kyseinen raportointityökalun

osa. Samalla kyseisen menuelementin teksti lihavoidaan, jotta käyttäjä tietää mitkä raportointityökalun osat ovat avoinna. (Kuva 10.)



Kuva 10. Raportointityökalu.

Jokainen raportointityökalun hakutoiminto on oma pieni sovelluksensa ja niitä voidaan lisätä ja poistaa näytöltä käyttäjän valintojen mukaan. Nämä pienoissovellukset on sijoitettu kukin oman div-elementtinsä sisään, jolla on yksilöllinen id-attribuutin arvo. Hakuvalintaosia voidaan tuoda ja poistaa näytöltä valikkoa klikkailemalla. Id-attribuutin perusteella sovelluksessa näytetään tai piilotetaan valittu osa JavaScriptin avulla. Jotta sovellus on saatu näyttävämmäksi, osien esiintuomisessa on käytetty JQuery funktiota show, jolle annetaan parametrina slow, jotta div-elementti tuodaan esiin hitaasti ikään kuin se liukuisi esiin. Samalla tavalla raportointityökalun osaa piilotettaessa käytetään funktiota slideUp, jolloin div-elementti näyttää ikään kuin liukuvan pikkuhiljaa ylös ja lopuksi katoavan näkyvistä. Kaikilla raportointityökalun osilla on oma kuvan 11 mukainen JQueryllä toteutettu kalenteri. Kalenteria tarvitaan, jotta voidaan valita aikaväli, josta kyseinen raportti, tilastotieto tai tapahtuma haetaan. Kalenterit päädyttiin toteuttamaan jokaiselle raportointityökalun osalle omansa, jotta tietoja voidaan hakea eri aikaväliltä samanaikaisesti eri työkalun osilla.

Kuva 11. Datepicker-kalenteri.

6.2 Liiketoimintaluokka

Liiketoimintaluokassa tehdään kaikki tiedon muokkaus ja analysointi. Liiketoimintaluokan konstruktori luo olion tietokantaluokasta, jonka kautta pystytään suorittamaan tietokantakyselyt. Liiketoimintaluokka on toteutettu niin, että kun käyttöliittymäluokka luo siitä olion, se saa parametrina tiedon avataanko testi- vai tuotantotietokantayhteys. Samalla tämä arvo asetetaan $tuotanto -muuttujan arvoksi, jotta muiltakin osin luokka tietää käyttää joko testi- tai tuotantotoimintoja. Toiminto on tehty, jotta sovellusta voidaan helposti testata ja poistaa virheet tarvittaessa testitietokantaa vasten.

Tietokantaa tutkittaessa ilmeni, että NextCall Pro toimii siten, että kun puhelinohjausjärjestelmään tulee puhelu tai sähköposti, siitä luodaan tapahtuma cases-tauluun. Tapahtumalle luodaan id, alkamisaika ja tieto, mihin sarjaan tapahtuma on osoitettu. Kullakin sarjalla on groups-taulussa puhelinnumero tai sähköpostiosoite, johon yhteydenotto tapahtuu. Tietokannassa on tieto, mihin sarjaan mis-

täkin numerosta tullut tapahtuma ohjataan. Sen jälkeen, kun tapahtuma on ohjattu oikeaan sarjaan, se siirtyy jonoon kunnes muut sitä ennen tulleet tapahtumat samasta sarjasta on vastaanotettu. Sarjoilla on myös eri kiireellisyysasteita, jonka perusteella puheluita ja sähköposteja ohjataan agenttisovelluksille. Esimerkiksi sähköpostisarjat on tehty siten, että kaikki puhelusarjat menevät niiden edelle kiireellisyydessä. Kyseisen tapahtuman vuoron tullessa ja vapaan agenttisovelluksen löydyttyä, tapahtuma poistuu jonosta ja siirtyy käsittelyyn agentille. Tässä kohtaa jonotusaika lasketaan ja se tallennetaan tietokantaan samoin kuin tieto, milloin tapahtuma alkoi, mikä agenttisovellus sen vastaanotti ja useita muita tämän työn kannalta tarpeettomia tietoja. Kantaan tallennetaan tieto myös siitä, jos asiakas keskeyttää puhelun kesken jonotuksen, agenttisovellus hylkää tapahtuman tai tapahtuu jokin virhe. Nämä ovat lähinnä raportoinnin kannalta olennaisia tietoja. Näissäkin tapauksissa kirjataan tieto agentista, ajankohdasta jne. Tapahtuman päätyttyä tallennetaan tieto päättymisajankohdasta, sekä tapahtuman kestosta.

6.2.1 Näyttömetodit

Edellä mainittujen tietojen löytymisen jälkeen ryhdyttiin toteuttamaan livescreen-sovellusta. Koska tiedetään, että tapahtuman tullessa tapahtumasta tallennetaan tieto ja ajankohta tietokantaan, pystyttiin rakentamaan tietokantakysely, jolla haetaan jonossa olevia tapahtumia. Tiedot haetaan tarkistamalla tietokannasta, montako tapahtumaa kyseisessä sarjassa on ilman käsittelyn aloitusaikaa. Pois piti sulkea mahdolliset virheelliset merkinnät, joten tarkistuksessa otettiin huomioon, että tapahtuman loppumisajankohtakenttä on tyhjä. Samaan tietokantatauluun tallennetaan myös muita tapahtumia, kuten esimerkiksi sovelluksesta ulospäin soitettuja puheluita, joten tietokantahaun tulee huomioida, että kyseessä on sisään tuleva puhelu.

NextCall Pro:ssa on itsessään toiminnallisuus, joka kertoo soittajalle arvion jonotusajasta. Arvioitu aika on kymmenen viimeisimmän puhelun keskiarvo. Tässä sovelluksessa haluttiin mahdollisimman reaaliaikainen tieto jonotusajan pituudesta, joten tieto haetaan kyseiseen sarjaan jonossa olevista puheluista pisimpään jo-

nossa olleen alkamisaika ja vertaillaan sitä tämänhetkiseen aikaan. Tietokannassa kaikki ajat ovat bigint-tyyppiä eli niistä pitää itse muodostaa ymmärrettäviä aikoja ja päivämääriä. Tieto on unix epoch -aika eli tieto siitä, montako sekuntia on kulunut unix-alkuajasta 1.1.1970 00:00:00. Aikojen muokkaus tehdään PHP-koodissa, jotta tietokannan rasitus pysyisi mahdollisimman pienenä. Tietokannasta haetaan vain pisimpään jonossa olleen tapahtuman bigint-aika eli suurin arvo, ja se muokataan koodissa oikeaan muotoon. Jonotusaikojen tarkistuksessa joudutaan sulkemaan pois kaikki perutut, keskeytetyt ja virhetilan saaneet tapahtumat, koska niillä ei välttämättä ole käsittelyn aloitusaikaa.

Jonotusaika haetaan vain, jos jonossa on puheluita. Tällä tavalla ei turhaan suoriteta tietokantakyselyä, joka ei palauta mitään, koska jos ei ole puheluita jonossa, jonotusaikaa ei ole vaan puhelu pääsee suoraan läpi. Jonotusajoissa oli lisäksi huomioitava sellainen seikka, että sarjoissa soitetaan ensiksi eripituisia tervehdys- ja informaatio-nauhoitteita. Jotta jonotusajat saadaan mahdollisimman paikkansa pitäviksi kussakin sarjassa, joudutaan jonotusajasta vähentämään nauhoitteen kesto.

Vastausprosentti kuudessakymmenessä sekunnissa on viestintäviraston tarkkailema arvo internetpalveluntarjoajien asiakaspalveluun ja vikailmoitukseen tulleista puheluista. Näin ollen tämä tieto halutaan näkymään tässä sovelluksessa, jotta sitä voidaan tarkkailla reaaliajassa. Vastausprosentti kuudessakymmenessä sekunnissa ja puhelumäärä haetaan siten, että haetaan kaikki tapahtumat, jotka ovat tulleet kuluvan vuorokauden aikana. Vastausprosentin laskemisessa tarvitaan myös tietoa puheluiden kokonaismäärästä, joten nämä kaksi kyselyä pystyttiin yhdistämään. ”vastausprosentti kuudessakymmenessä sekunnissa” -haussa tulee sivutuotteena, tieto siitä, paljonko puheluita on tullut kuluvana vuorokautena. Puhelumäärään tarvitaan siis tieto kaikista tiettyyn sarjaan tulleista puheluista kuluvan vuorokauden aikana ja ”vastausprosenttiin kuudessakymmenessä sekunnissa” tieto siitä, moneenko näistä puheluista on vastattu kuudenkymmenen sekunnin sisällä. Haussa tarkistetaan vain, että puhelu on tullut kyseiseen sarjaan kuluvan vuorokauden 00:00:00-24:00:00 välisenä aikana. Haussa huomioidaan, että kyseessä on tuleva

puhelu ja, että se on tullut palveluaikana, jos kyseessä on sarja, joka ei ole auki koko ajan.

Puheluita otetaan vastaan sarjoissa eri aikoihin. Vikailmoituksia otetaan vastaan kaikkina vuorokaudenaikoina vuoden jokaisena päivänä, kun taas toiset sarjat ovat avoinna vain arkisin esimerkiksi 08:00-21:00 välisenä aikana. Tästä johtuen tietoja haettaessa pitää tarkistaa service_calendar -taulusta, että kyseinen sarja on avoinna puhelun tullessa, sillä tapahtumasta tallennetaan alkutiedot NextCall Prossa jo siinä vaiheessa, kun puhelu on tulossa ja sitä lähdetään ohjaamaan oikeaan sarjaan. Tällöin myös aukioloajan ulkopuoliset puhelut tallentuvat tietokantaan, mutta niitä ei saa ottaa huomioon vastausprosentteja laskettaessa. Tapahtuma saatetaan siirtää eri sarjaan kuin minne se alun perin on tullut, joten tämä mahdollisuus on huomioitava kyselyitä laadittaessa. Tietokannasta löytyi oma kenttä tapahtuman alkuperäisestä sarjasta sekä tapahtuman hoitaneesta sarjasta.

6.2.2 Raportointityökalu metodit

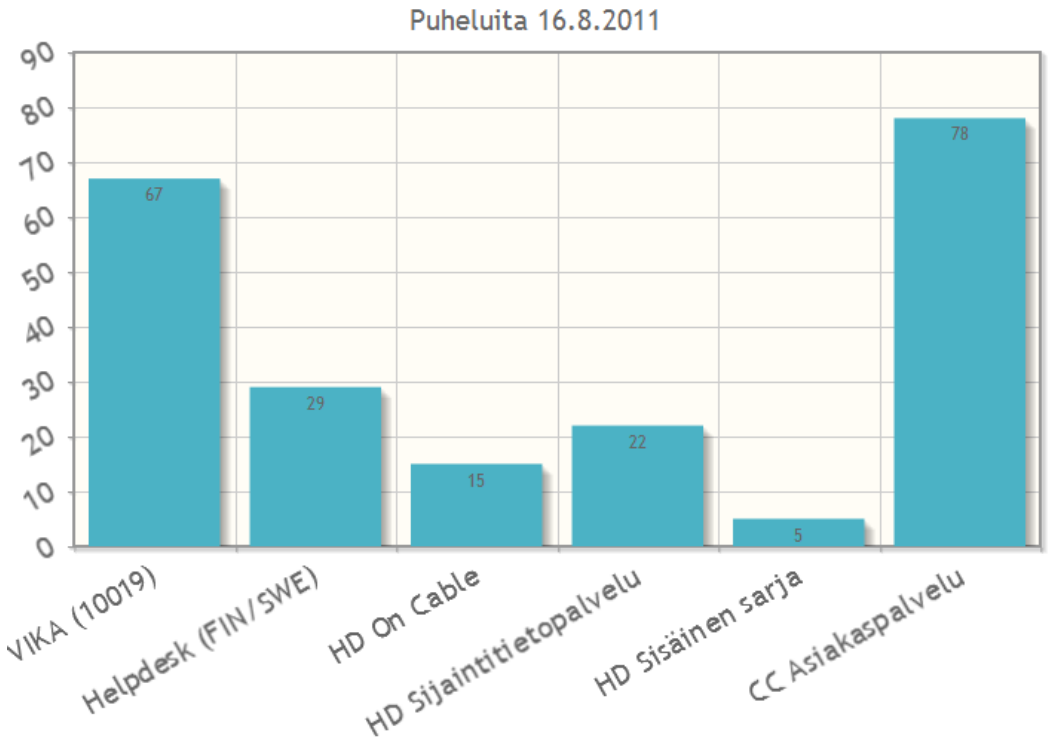
Raportointityökalua toteutettaessa pystyttiin osittain käyttämään samaa liiketoimintaluokkaa kuin livescreen-sovelluksessa. Esimerkiksi sisäänkirjautumisessa käytetään kummassakin sovelluksessa samaa liiketoimintaluokkaa. Käyttäjän valitsemien sarjojen haku MySQL-tietokannasta on myös ohjelmoituna ainoastaan livescreen-sovelluksen liiketoimintaluokkaan. Kumpikin liiketoimintaluokka käyttää yhteistä tietokantaluokkaa. Ainoana erona on, että livescreen-sovelluksessa yhteys luodaan tuotantotietokantaan ja raportoinnissa varapalvelimen tietokantaan. Asiasta kerrotaan tarkemmin kappaleessa 6.3.

Ensimmäinen raportointityökalun osa, joka toteutettiin, oli puhelutietohistoriahakutoiminto. Tämä toiminto oli suhteellisen helppo toteuttaa, koska siinä näytetään samantyyppinen taulukko kuin livescreen-sovelluksessa. Erona vain se, että näytettävää tietoa on muutama kenttä enemmän ja tiedot haetaan käyttäjän valitsemalta aikaväliltä. Puhelutietohistoriahakutoiminto käyttää hyväkseen livescreen-sovelluksen liiketoimintaluokkaa. Raportointityökalun liiketoimintaluokasta luodaan olio livescreen-sovelluksen liiketoimintaluokasta ja näin voidaan käyttää

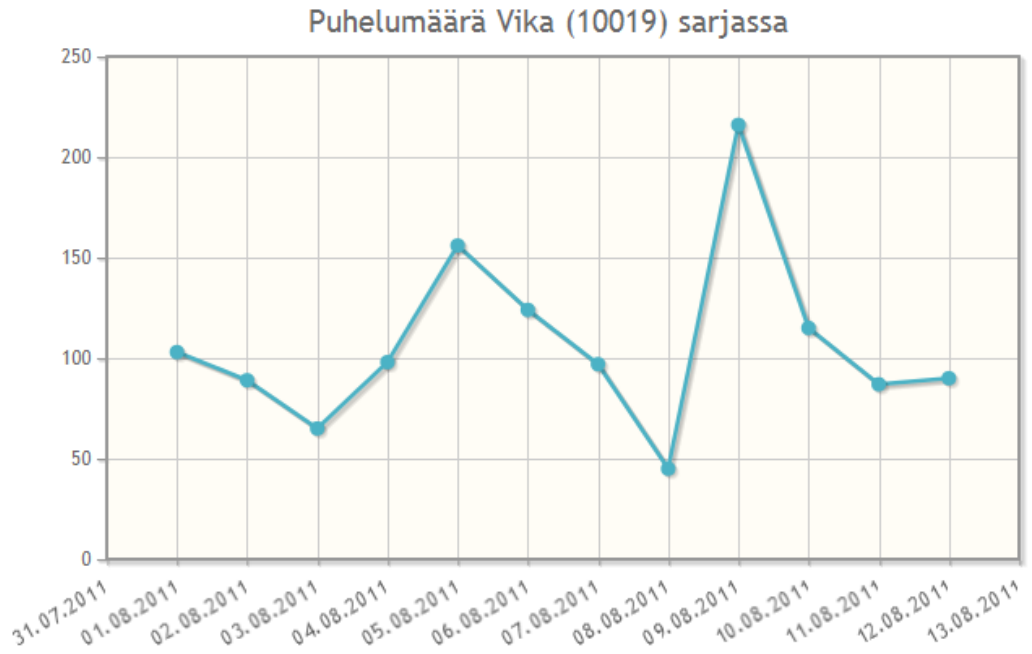
samoja metodeja tiedon hakuun kuin livescreen-sovelluksessa. Käyttäjä saa itse valita, mistä sarjoista tieto näytetään. Oletuksena näytetään tiedot niistä sarjoista, jotka käyttäjällä on valittuna livescreen-sovelluksessa näytettäviksi.

Raportointityökalun numerohakuosalla haetaan tietoa siitä, onko asiakas ollut aikaisemmin yhteydessä ja tieto siitä kuka tapahtumaa on hoitanut. Aikaisempia tapahtumia voidaan hakea joko puhelinnumeron, sähköpostiosoitteen tai hoitaneen agentin perusteella. Tapahtumien haku tehdään käyttäjän valitsemalta aikaväliltä. Haku voidaan myös suorittaa tietyltä aikaväliltä ilman muita hakukriteerejä. Tämä on tarpeellista, jos tapahtumasta ei tiedetä muuta kuin arvioitu aika. Hakutuloksena saadaan lista tapahtumista, jotka vastaavat hakukriteerejä. Tuloksessa näytetään numero tai sähköpostiosoite, josta tapahtuma on tullut, kuka tapahtumaa on hoitanut, aloitusaika, mihin sarjaan tapahtuma on tullut, puheluissa puhelun kesto ja linkki, josta voi aloittaa puhelun kuuntelun. Peruskäyttäjällä on oikeus kuunnella vain omia puheluitansa. Kuuntelun avauslinkki näytetään vain, jos tietokannasta saatu tapahtumaa hoitanut agentti vastaa $_SESSION muuttujan ”kayttaja”-kentän arvoa. Tämä arvo asetetaan sisäänkirjautumisen yhteydessä. Sovelluksen ylläpito- eli admin-käyttäjä voi kuunnella kaikkien käyttäjien puheluita. Ylläpitokäyttäjiä ovat valvomokoordinaattori ja esimiehet. Käyttäjän rooli saadaan selville samasta tietokantataulusta, kuin mistä itse kirjautumistiedot löytyvät.

Tilasto-osalla tuotetaan sarjoista valitulta aikaväliltä tilastoja, joista selviää, mikä on ollut ruuhkaisin aika vuorokaudesta jokaiselta aikavälin päivältä, tai nähdään, kuinka puhelut ovat ajoittuneet eri sarjoissa eri ajankohdille. Tilasto-osan tulokset näytetään kuvien 11 ja 12 mukaisina jqPlot-diagrammeina.



Kuva 11. Tilasto-osan puhelumäärä diagrammi.



Kuva 12. Puhelumäärä-diagrammi Vika (10019) sarjasta elokuun alkupuoliskolta.

jqPlot on jQueryn lisäosa, jolla voidaan piirtää diagrammeja. jqPlot-diagrammeihin on mahdollista lisätä vuorovaikutteista toimintaa. Kuvien 11 ja 12 kalenterissa, jos käyttäjä vie osoittimen kuvan 11 palkkien tai kuvan 12 pisteiden päälle, näytetään pieni ruutu, joka kertoo palkin tai pisteen arvot. Tietokantaluokalta saadut tietokannasta haetut arvot, jotka ovat PHP-objektina, välitetään jqPlot-oliolle JSON-enkoodattuna taulukkona. JSON (JavaScript Object Notation) on kieliriippumaton tekstimuoto, jota lähes kaikki ohjelmointikielet ymmärtävät. PHP-koodissa tiedot muutetaan JSON-taulukoksi PHP:n sisäisellä json_encode-funktiolla. JSON-enkoodattu taulukko välitetään jqPlot-funktiolle, joka tekee siitä ohjelmoijan määritysten mukaisen diagrammin.

Agenttikohtaisella statistiikkaosalla haetaan tietoa yksittäisen agentin vastaanottamista tapahtumista. Käyttäjä valitsee aikavälin, jolta tiedot haetaan sekä agentin, jonka tiedot haetaan. Aikaväli valitaan kahdella jQuery-kirjaston datepicker-oliolla. Agentti valitaan alasvetovalikosta, johon on haettuna tietokannasta kaikki NextCall Pro käyttäjät eli agentit. Käyttäjän valitessa agentin, eli kun alavetovalikon valittu alkio vaihtuu, alasvetovalikosta onchange-attribuutilla lomake lähetetään sovellukselle itselleen JavaScript-funktiolla submit. Sovelluksen PHP-koodissa tutkitaan onko $_POST muuttujassa asetettuna arvot alkuaika, loppuaika ja agentti. Jos näissä kentissä on hyväksytyt arvot, haetaan niiden perusteella tietokannasta tiedot valitulta aikaväliltä valitulle agentille. Syötetiedoista tarkistetaan, että ajat ovat aikamuodossa ja, että alkuaika on pienempi kuin loppuaika. Agentille haetaan tiedot, moneenko puheluun hän on vastannut helpdesk-sarjoissa ja moneenko vika-sarjoissa sekä keskimääräinen puheluiden kesto näissä sarjaryhmissä. Molempien sarjaryhmien sarjat on ennalta määrätty sovelluksen tilaajan toimesta.

Puheluiden jakauma raportointityökalun osa on Anvialla hoidettavan ulkoisen asiakkaan sarjoja varten tehty raportointiosa. Puheluita tulee sarjaan kahdesta eri numerosta, mutta NextCall Pro ohjaa ne molemmat samaan sarjaan. Anvian val-

vomon esimiehillä on tarve tietää kummasta numerosta tähän sarjaan tullut puhelu on tullut. Tässä osassa aikavälivalinta poikkeaa hieman muiden osien aikavälivalinnasta. Päivämäärät valitaan samaan tapaan kuin muissakin, mutta kellonaika hakuvälillä valitaan se aikaväli, jolta tiedot haetaan päivittäin. Käyttäjän rajatessa aikaväliksi kellonajat 10:00–21:00 haetaan puheluiden jakaumatieto valittujen päivien valitulta aikaväliltä. Käyttäjän valittua aikaväli, lähetetään lomake sovellukselle itselleen samaan tapaan kuin muissakin raportointityökalunosissa. Tietokannasta haetaan valintoja vastaavat tietueet ja näytetään käyttäjälle. Käyttäjälle näytetään kaksi tulostaulukkoa eli molempia ulkoisen asiakkaan numeroita vastaava taulukko erikseen. Taulukoissa näytetään tiedot soittoajasta, numerosta johon puhelu on tullut, puhelun kesto sekä linkki, josta aloitetaan puhelun tallenteen kuuntelu. SSP-puheluiden jakaumaosa näkyy ainoastaan admin-ryhmään kuuluville käyttäjille, joten ei tarvitse tehdä erillistä tarkistusta, onko käyttäjällä oikeus kuunnella puhelu.

6.3 Raportointityökalu

Raportointityökalussa päädyttiin käyttämään NextCall Pro:n varapalvelimen eli varmuuskopiotietokantaa. Varapalvelimen tietokanta on kopio tuotantokäytössä olevasta tietokannasta. Kopiointi tapahtuu joka yö klo 00:00 alkaen, jolloin puheluiden määrä ja näin ollen myös tietokannan käyttöaste on pienimmillään. Varapalvelimen tietokannan käyttöön päädyttiin lähinnä siitä syystä, että tuotantotietokanta ei kuormitu turhaan. Näin ollen raportointityökalun tietokantahaut eivät ole aivan niin tehokkuusriippuvaisia kuin livescreen-sovelluksen. NextCall Prossa on agenttisovelluksessa itsessään historiatietoja hyvin näyttävä toiminnallisuus. Tiedot saa haettua kätevästi vain agenttisovelluksen käynnistyksestä alkaen eli samalta päivältä. Tästä syystä riittää, että raportoinnin osalta saadaan haettua tietoa edellisestä päivästä lähtien eli juuri mitä varapalvelimen tietokannasta saadaan.

6.4 Tietokantaluokka

Tietokantaluokka hoitaa yhteyden tietokantaan. Koska tässä sovelluksessa käytetään PDO-driver -luokkaa, hoitaa se käytännössä tietokantayhteyden eli kun liike-

toimintaluokka luo tietokantaluokasta olion luodaan siellä PDO-olio. Oliota luotaessa sille annetaan parametrina tietokannan tyyppi, -osoite ja -portti, johon otetaan yhteyttä sekä tietokannannimi, -käyttäjätunnus ja -salasana.

Koska PDO-olion luonti ja näin ollen tietokantayhteyden luonti on tehty omaan luokkaansa, voidaan siitä luoda sovelluksessa kaksi oliota. Toinen olio huolehtii yhteydestä tuotantotietokantaan livescreen-sovellusta varten ja toinen huolehtii yhteydestä varapalvelimen tietokantaan raportointityökalua varten. Tietokantaluokka päättelee kumpaan tietokantaan yhteys luodaan parametrina saamansa tuotanto muuttujan arvon perusteella. Arvon ollessa ’true’ eli tosi otetaan yhteys tuotantotietokantaan, muutoin yhteys luodaan varapalvelimen tietokantaan.

Tietokantaluokka huolehtii myös lokitapahtumien tallennuksesta. Kutsumalla tietokantaluokasta luodun olion getlogger-metodia avataan yhteys lokitiedostoon. Tässä työssä käytetään lokitiedostona tavallista tekstitiedostoa. Joka kuukaudelle luodaan oma lokitiedostonsa, jotta tiedoston tekstimäärä ei kasva liian suureksi ja jotta vanhoja lokeja on helppo tarvittaessa poistaa tai arkistoida. Lokitiedostoon kirjoitetaan tietoa aina, kun sovelluksessa tapahtuu jokin virhe. Virheen tapahtuessa sekä käyttäjälle että lokitiedostoon annetaan virheilmoitus. Käyttäjälle annetaan suurpiirteisempi virheilmoitus ja lokitiedostoon kirjoitetaan mahdollisimman paljon tietoa tapahtuneesta virheestä sekä ajankohta, milloin virhe on tapahtunut. Tämä helpottaa sovelluksen vikatilanteiden tarkastelua jälkikäteen.

7 JOHTOPÄÄTÖKSET JA POHDINTA

Opinnäytetyön tuloksena saatiin toteutettua näyttö, jolla voidaan Anvian valvomossa ja helpdeskissä seurata sarjojen tilanteita reaaliajassa. Lisäksi toteutettiin raportointisovellus, jolla voidaan hakea erilaisia tilastotietoja Anvian johdolle ja viestintävirastolle sekä itse NextCall Pron käyttäjille. Näiltä osin opinnäytetyön tavoitteet toteutuivat. Näyttösovelluksesta saatiin tarpeeksi kevyt ja tarpeeksi vähän tietokantaa kuormittava, jotta Capricoden osalta saatiin lupa, että sitä voidaan käyttää ilman, että se kuormittaa tietokantaa liikaa. Raportointityökaluun voidaan helposti lisätä uusia ominaisuuksia, koska se on toteutettu siten, että jokainen raportointityökalun osa on oma pieni sovelluksensa. Todennäköisesti näitä sovelluksen osia tullaan tekemään vielä lisää sitä mukaa, kun tarvetta ilmene uusille ominaisuuksille.

Tietokantakyselyiden tehostaminen oli suurimpia ongelmia työn toteutuksen osalta. Testaamalla useita eri variaatiota kyselyistä ja käyttämällä tässä työssä esiteltyjä menetelmiä saatiin tehokkaat ja tietokantaa mahdollisimman vähän rasittavat kyselyt tehtyä. Tuotantotietokantaan tehtyjen testien perusteella tietokannassa tai palvelimella, jolla tietokanta sijaitsee, ei havaittu mitään hälyttävää sovellusta testattaessa. Näin ollen tämä osa työstä onnistui hyvin.

Tietoturvallisuuden osalta sovelluksessa pyrittiin ottamaan huomioon kaikki tässä työssä esitellyt tietoturvauhat. Tietoturvauhkia on todellisuudessa vielä enemmän kuin tässä työssä esitellyt ja niitä tulee kokoajan lisää, joten tietoturvallisuuden osalta työssä mahdollisesti löytyy joiltain osin parannettavaa. Autentikointimenetelmänä HTTP Basic Authentication esimerkiksi ei ole kaikista varmimpia menetelmiä.

Kolmitasomallin ja sovelluksen osissa toteuttamisen idea toteutui ja toimi tämän työn osalta erinomaisesti. Käyttöliittymä luokat saatiin toteutettua miltei valmiiksi jo ennen kuin sovelluksella saatiin edes tietokantaan yhteyttä. Tietokantayhteyden muodostaminen NextCall Pro:n PostgresSQL-tietokantaan oli yksi suurimmista työn toteutuksessa esiin tulleista ongelmista. Aluksi ongelmana oli palomuuri, jo-

ka esti yhteyden palvelinten välillä. Tämä korjattiin avaamalla reitti palomuuriin. Työn suunnittelu- ja toteutusvaiheessa tietokantana käytettiin varapalvelimen tietokantaa ja, kun siirryttiin testivaiheessa tuotantotietokantaan, ongelmana oli jälleen yhteyden saaminen. Palomuurista avattiin reitti samaan tapaan kuin varapalvelimenkin tietokantaan, mutta yhteyttä ei siltikään saatu muodostettua. Tutkimalla PDO-driver olion palauttamia virheilmoituksia selvisi, että syynä on PostgreSQL-tietokannan ominaisuus, joka estää TCP/IP pyyntöjen vastaanoton mikäli sitä ei sallita postgresql.conf-tiedostossa.

Opinnäytetyön teon edetessä opin uutta PHP-ohjelmoinnista ja erityisesti tietoturvallisuudesta PHP- ja yleisesti web-ohjelmoinnissa. Ennen työn aloitusta en ollut käyttänyt jQuery:ä. Tätä työtä tehdessä opin paljon jQuery:n käytöstä websovellusten monipuolistamisessa. Etsin kaiken sovelluksissa käytetyn tiedon PostgresSQL-tietokannasta ja opin sen myötä relaatiotietokantojen ja erityisesti PostgresSQL-tietokannan toiminnasta ja SQL-kyselyiden tekemisestä sekä niiden tehostamisesta.

Työn tuloksena toteutettuihin sovelluksiin olen kohtalaisen tyytyväinen. Raportointityökaluosassa toteutuu kaikki määrityksissä vaaditut kohdat sekä muutamia toteutuksen aikana esille tulleita tarpeita. Lisäksi sovellukseen on helppo lisätä uusia toiminnallisuuksia, koska osat on toteutettu omina pieninä kokonaisuuksinaan. Tilaajan taholta on tullut jo muutamia ehdotuksia lisäominaisuuksista. Lisäominaisuudet toteutetaan seuraavaan versioon sovelluksesta. Näyttö olisi saanut olla mielestäni ulkonäöltään ja suorituskyvyltään parempi. Työn määrityksissä tietojen päivitysvälitavoite näytöllä oli 1 minuutti. Mielestäni se on liian pitkä aika. Päivitysväliä ei voi laittaa pienemmäksi, jotta palvelimen ja tietokannan kuormitus ei nouse liian suureksi. Ulkonäöltään sovelluksen ei tarvinnut olla silmiä hivelevä vaan mahdollisimman selkeä ja helposti luettava. Mielestäni näyttö on selkeä ja yksinkertainen, mutta jollain lailla se jää latteaksi.

LÄHTEET

/1/ Alshanetsky, I. 2005. php|architect´s Guide to PHP Security. First Edition. Toronto. Marco Tabini & Associates, Inc.

/2/ Anvia. Konsernin rakenne. Viitattu 13.4.2011. http://www.anvia.fi/fi-FI/Konserni/tietoakonsernista/konsernirakenne/Sivut/default.aspx

/3/ Anvia. Puhelinyhtiöstä viestinnän moniosaajaksi. Viitattu 13.4.2011. http://www.anvia.fi/fi-FI/Konserni/tietoakonsernista/historia/Sivut/default.aspx

/4/ Asleson, R. Schutta, N. 2006. Ajax - Tehokas hallinta. Jyväskylä. Gummerus kirjapaino Oy Readme.fi.

/5/ Capricode Oy. Capricode – Company History Viitattu 14.4.2011. http://www.capricode.com/company

/6/ Greenspan, J. Bulger, B. 2001. MySQL/PHP Database Applications. New York. M&T Books.

/7/ Hovi, A. 2004. SQL-opas. 1.painos. Saarijärvi. Docendo Finland Oy.

/8/ Howard, M. Leblalanc, D. 2002. Ohjelmoijan tietoturvaopas. 2004. Helsinki. Edita prima Oy.

/9/ Kontio, M. Vierimaa, P. Niskanen, P. 2001. www-ohjelmointi. 2. painos. Jyväskylä. Oy Edita Ab.

/10/ Kolehmainen, K. 2006. PHP & MySQL - Teoriasta käytäntöön. 1.painos. Jyväskylä. Gummerus kirjapaino Oy Readme.fi.

/11/ Longman, A. 1998. A history of HTML. Viitattu 7.8.2011. http://www.w3.org/People/Raggett/book4/ch02.html

/12/ Nguyen, T. 2003. Cascading Style Sheets. Viitattu 3.9.2011. http://www.wellesley.edu/Computing/Dreamweaver/CSS/cssMain.html

/13/ PostgreSQL Global Development Group. About. Viitattu 5.8.2011 http://www.postgresql.org/about/

/14/ PostgreSQL Global Development Group. EXPLAIN. Viitattu 10.8.2011 http://www.postgresql.org/docs/8.1/static/sql-explain.html

/15/ PostgreSQL Global Development Group. History. Viitattu 5.8.2011 http://www.postgresql.org/about/history

/16/ Rantala A. 2002. PHP : web-ohjelmoinnin peruskirja. 1.painos. Porvoo. Docendo Finland Oy.

/17/ Rantala A. 2005. Web-ohjelmointi. 1.painos. Jyväskylä. Docendo Finland Oy.

/18/ The jQuery Project. How jQuery Works. Viitattu 12.8.2011. http://docs.jquery.com/How_jQuery_Works

/19/ the PHP Group. 2011. History of PHP. Viitattu 6.8.2011 http://php.net/manual/en/history.php.php

/20/ the PHP Group. 2011. PHP Data Objects. Viitattu 9.8.2011. http://php.net/manual/en/book.pdo.php

/21/ the PHP Group. 2011. PHP Licensing. Viitattu 6.8.2011. http://fi2.php.net/license/

/22/ the PHP Group. 2011. What can PHP do?. Viitaattu 7.8.2011. http://fi2.php.net/manual/en/intro-whatcando.php

/23/ Zandstra, M. 2001. PHP Trainer Kit. Helsinki. Oy Edita Ab.

/24/ Zend-The PHP Company. PHP and Zend Engine. Viitattu 6.8.2011. http://www.zend.com/en/community/php/