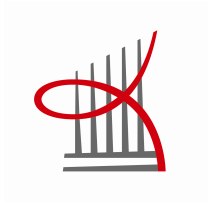


TIEDONKERÄYSJÄRJESTELMÄ

Teemu Perämäki

Opinnäytetyö
Toukokuu 2013
Tietotekniikka
Ohjelmistotuotanto

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikka
Ohjelmistotuotanto

TEEMU PERÄMÄKI:
Tiedonkeräysjärjestelmä

Opinnäytetyö 20 sivua, joista liitteitä 0 sivua
Toukokuu 2013

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa tiedonkeräysjärjestelmä, jolla kerätään automatisoidusti usealta palvelimelta tietoja ja tehdään näistä laskutusohjelmistoon vietävissä oleva tiedosto.

Järjestelmä tulee olla käytettävissä alustasta riippumatta internetselaimen kautta.

Ohjelmisto päätettiin toteuttaa käyttäen PHP-ohjelmointikieltä. Järjestelmä kerää tiedot usealta palvelimelta, laskee haetuista tiedoista hinnat ja muodostaa näistä laskutusohjelmistoon vietävän tiedoston. Järjestelmä myös merkitsee kyseiset pelit ja kuukaudet laskutetuiksi, ja päättelee mitkä peleistä on mahdollisia testipelejä.

Asiasanat: PHP, MySQL, tiedonkeräys, Apache

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in Information Technology
Software Engineering

TEEMU PERÄMÄKI:
Data collection system

Bachelor's thesis 20 pages, appendices 0 pages
May 2013

The purpose of this project was to plan and produce a data collection system, which collects data automatically from multiple servers and builds a file that can be then uploaded to invoicing system.

System needed to be usable through web browser on all platforms.

We decided to make the system using PHP programming language. System collects the data from multiple servers, calculates the price from this data and builds a file that is uploadable to the invoicing system. System also marks these games and months as invoiced, and also determines which games are possibly testgames.

Keywords: PHP, MySQL, data collection, Apache

SISÄLLYS

ERITYISSANASTO

MySQL	Relaatiotietokanta
PHP	Ohjelmointikieli, jota käytetään dynaamisien sivustojen luomiseen
Apache	Palvelinohjelmisto sivustojen ylläpitämiseen

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa tiedonkeräysjärjestelmä, jolla voidaan tiedot kerätä helposti usealta palvelimelta laskutusta varten.

Tarve syntyi kun yrityksen asiakasmäärä lähti kasvuun, ja samalla palvelinten määrä kasvoi huomattavasti. Koska laskutukseen tarvitaan tiedot palvelinten käytöstä, alkoi laskun tekemiseen tarvittava työmäärä kasvaa liian suureksi manuaalisesti tehtynä.

Seuraavissa kappaleissa käydään läpi tiedonkeräystä yleisesti, projektiin valittuja ympäristöjä ja suunnittelua. Myöhemmissä kappaleissa tutkiskellaan eri tekniikoiden ja käyttöliittymän toimintaa.

2 TIEDONKERÄYS YLEISESTI

Tässä kappaleessa käsitellään tiedonkeräystä yleisellä tasolla.

”Tiedonkeräyksen tarkoitus on hankkia tietoa joko arkistoitavaksi, päätöksenteon avuksi, tai muille välitettäväksi.” (Data Collection, 2013)

2.1 Tiedon säilöminen

Kerätyn tiedon säilyttämisessä apuna käytetään usein tietokantoja. Usein käytetty tietokantatyyppi on relaatiotietokanta. Relaatiotietokannassa tietotaulujen välille luodaan yhteyksiä. Yhteydet ovat viittauksia toisen taulun pääavainkenttään. Esimerkiksi, jos yhdessä taulussa on listattuna kirjailijoita, ja kirjailija jonka pääavaimena on numero 1, on nimeltään Kalle Kirjailija. Toisessa taulussa on listattuna kirjoja, ja tämän toisen taulun yksi sarake on niin sanottu vierasavain, eli kenttä joka sisältää toisen taulun pääavaimen. Nyt jos esimerkiksi kirjoja sisältävässä taulussa olisi tietueet kirjoille X ja Y, ja näiden vierasavain sarakkeissa olisi arvo 1, tarkoittaisi se, että ne ovat kirjailijan Kalle Kirjailija kirjoja. (Tietokanta, 2013)

Tietokannassa on usein kahdenlaista tietoa. Staattista tietoa, joka muuttuu harvoin, ja tapahtumatietoa, joka muuttuu jatkuvasti. Yleensä staattista tietoa on suhteellisen vähän, mutta tapahtumatietoa voi olla miljoonia rivejä. Esimerkkinä kirjakauppa, jossa staattisena tietona toimii asiakas, kirja, julkaisija ja toimittaja. Tapahtumatietona toimii esimerkiksi myyntitapahtumat. Näiden tietotyyppien käyttö on usein myös hyvin erilaista. Esimerkiksi asiakastietoja haettaessa usein haetaan jonkun yksittäisen asiakkaan tietoja, eli hakutulos on hyvin pieni, mutta tapahtumatietoja haettaessa haetaan esimerkiksi viime vuoden myynnit, jolloin hakutulos on hyvin suuri. (Powell 2005, 171-175.)

Tähtimalli on yksi esimerkki tavasta tehdä tietokannasta selkeä ja nopea. Tähtimalli perustuu siihen, että staattiset tiedot pidetään omissa tauluissaan, ja niihin viitataan tapahtumataulusta. Viittaukset ovat yleensä ”One to many” -viittauksia, jolloin siis esimerkiksi yhdellä myyntitapahtumalla voi olla vain yksi asiakas, mutta yhdellä asiakkaalla voi olla useita myyntitapahtumia. (Powell 2005, 176-177.)

3 YMPÄRISTÖ

Tässä kappaleessa käydään läpi ohjelmistovalintoja ja sitä miksi ne tulivat valituksi.

3.1 PHP

PHP-ohjelmointikieli tuli valituksi pääosin sen takia, että se oli entuudestaan hyvin tuttu kieli. Valintaan vaikutti myös PHP-ohjelmointikielestä löytyvät tietokantayhteydet, joita hyödyntäen siis yhdistäminen MySQL- ja PostgreSQL-tietokantoihin käy kivuttomasti.

Kielen tuli myös mahdollistaa sivujen dynaamisuus, eli sisällön on voitava muuttua tietokannan sisältämien tietojen perusteella.

3.2 MySQL

Tietokannaksi valikoitui MySQL sen ollessa entuudestaan tuttu, mutta valintaan vaikutti myös se, että MySQL on suosittu muiden kehittäjien keskuudessa, joten siihen liittyvissä ongelmissa on helppo löytää erilaisia esimerkkejä ja ratkaisuja ongelmiin.

3.3 Apache

Palvelinohjelmistoksi valikoitui Apache. Apache tulee esiasennettuna monien Linux-palvelinten mukana, joten se oli luonteva valinta. Apachen konfigurointi käyttötarkoitukseen sopivaksi vaati kuitenkin PHP- ja SSL-lisäosien asennuksen. Vaatimalla suojattu yhteys ohjelmiston käyttämiseksi saavutettiin hieman parempi tietoturva.

4 SUUNNITTELU

Tässä kappaleessa kuvaillaan suunnitteluvaiheen etenemistä.

4.1 Toiminnalliset vaatimukset

Suunnitteluvaiheessa toiminnallisista vaatimuksista päätettäessä päädyttiin siihen, että ohjelman tulee kyetä

- keräämään tiedot usealta eri palvelimelta yhtäaikaisesti
- hinnoittelemaan eri hinnastojen perusteella
- tunnistamaan testipelit
- pitämään kirjaa jo laskutetuista peleistä ja kuukausimaksuista

4.2 Käyttöliittymän suunnittelu

Käyttöliittymä suunniteltiin siten, että ohjelmiston hallintaan liittyvät, harvoin käytetyt toiminnallisuudet sijoitettiin erilleen normaaleista, usein käytetyistä toiminnallisuuksista.

5 TEKNIIKAT

Tässä kappaleessa käydään läpi käytettyjä tekniikoita ohjelman eri osa-alueilla.

5.1 Tiedonkeräys

Tiedonkeräys alkaa, kun käyttäjä painaa käyttöliittymästä ”Get Data” -painiketta. Tällöin järjestelmä lukee tietokannasta listan palvelimista, joilta sen tulee tiedot kerätä. Järjestelmä lähettää suojatun POST -pyynnön jokaiselle palvelimelle, joka sisältää tarvittavat tunnistautumistiedot. Mikäli tunnistautuminen onnistuu, hakee palvelin tietokannasta tarvittavat tiedot ja muuntaa ne JSON -muotoiseksi. Kun muunnos on suoritettu, vastaa palvelin POST -pyyntöön välittäen tiedot järjestelmälle. Tiedon välittäminen takaisin järjestelmälle tapahtuu tässä tapauksessa siten, että palvelin, joka on tietoa lähettämässä takaisin, tulostaa tiedot JSON -muotoisena sivulle, johon tietoa keräävä palvelin pyynnön lähetti. Tietoa keräävä palvelin hakee tämän sivun sisällön kokonaisuudessaan, ja tarkistaa onko saatu tieto oikeassa muodossa. Mikäli järjestelmän mielestä tieto voidaan hyväksyä, alkaa se käsittelemään sitä. Järjestelmä purkaa JSON -muotoiset tietueet ja muuntaa ne tietokantaan soveltuviksi.

5.2 Tiedon vienti laskutusohjelmistoon

Laskutusohjelmistoon vietävän tiedoston luomiseksi tulee käyttäjän määritellä palvelin, laskutettavien kuukausimaksujen määrä ja aikaväli, jolta pelit laskutetaan. Myös hinnaston ja valuutan tulee olla asetettuna. Käyttäjän määriteltyä nämä tiedot näytetään yhteenveto laskutettavista tietueista. Käyttäjän hyväksyttyä nämä tietueet, luodaan palvelimelle CSV -tyyppinen tiedosto, jonka lataus aloitetaan välittömästi. Nämä tiedostot myös tallennetaan suojattuun sijaintiin palvelimella varmuuskopioina.

5.3 Pelatun pelin sijainti

Pelattujen FlagHunt -pelien sijaintia on mahdollista tarkastella pelilistalta linkkiä painamalla. Kun käyttäjä painaa linkkiä, avataan uusi välilehti, jossa lähetetään GET -kutsu Google Maps -palvelulle. GET -kutsun mukana lähetetään sijaintitiedot pelatusta pelis-

tä. Tämä avaa Googlen Maps -palvelun selaimeen, jossa näkyvään karttaan on merkattu vihreällä nuolella pelatun pelin sijainti.

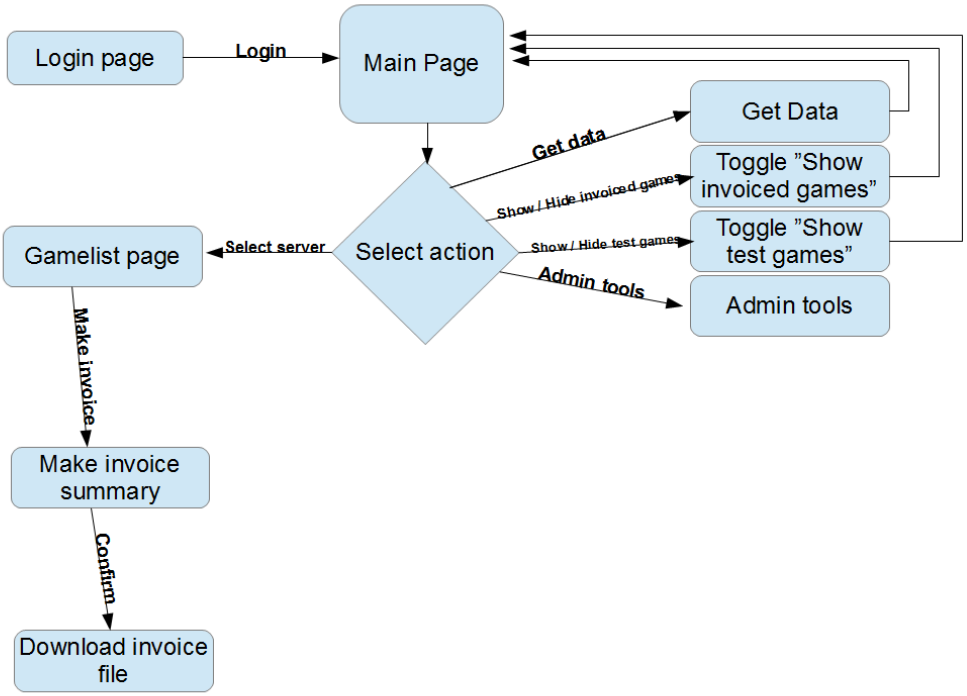
5.4 Hinnoittelu

Laskutusohjelmistoon vietävään tiedostoon lasketaan järjestelmän toimesta hinnat jokaiselle tietueelle valmiiksi. Hintojen laskenta perustuu hinnastoon, joita voidaan luoda järjestelmän asetukset -välilehden kautta. Jokaiselle palvelimelle määritellään, että mitä hinnastoa sen tulee käyttää laskutustietoja luodessa. Tämä mahdollistaa useiden eri hinnastojen käytön eri palvelimilla, tai vaihtoehtoisesti yleisen hinnaston käytön useilla palvelimilla. Hinnastoon asetetaan myös valuutta, jossa hinnat on syötetty hinnastoon. Valuutta tulee luoda ennen hinnastoa, ja valuutalle tulee antaa selkokielinen kuvaus, sekä laskutusjärjestelmästä saatava valuuttatunnus.

Hinnan laskemiseen käytetään myös pelissä olleiden pelaajien määrää. Sillä haettavasta tiedosta selviää ainoastaan pelissä olleiden puhelimien määrä, tulee hinnastoon olla asetettuna myös puhelinkohtainen pelaajamäärä. Esimerkiksi jos pelissä on ollut puhelimia 10 kappaletta, ja puhelinkohtainen pelaajamäärä on hinnastossa 2, niin laskutustietoon merkataan pelaajamääräksi 20 pelaajaa.

6 KÄYTTÖLIITTYMÄ

6.1 Käyttöliittymän virtauskaavio



KUVA 1. Käyttöliittymän virtauskaavio

Kuvassa 1 on kuvattu käyttöliittymän siirtymät eri näkymistä toisiin. Kuvasta on selkeyden vuoksi jätetty pois uloskirjautumiseen käytettävä painike, joka on käytettävissä kaikista näkymistä käsin. Uloskirjautumisen jälkeen käyttäjä ohjataan takaisin sisäänkirjautumisikkunaan.

Pääsivulta valittavista toiminnoista ”Get Data” ja molemmat ”Toggle” -toiminnot palauttavat käyttäjän takaisin päänäkymään samaan kohtaan, jossa käyttäjä oli painiketta painaessa, kun kyseinen toiminto on järjestelmän puolesta suoritettu ja mahdolliset muutokset näkymään tehty.

”Toggle” -toiminnoilla on mahdollista näyttää tai piilottaa listalta jo laskutetut pelit, sekä testipeleiksi merkityt pelit.

6.2 Sisäänkirjautuminen

Sisäänkirjautumisikkuna on ensimmäinen käyttäjälle näkyvä käyttöliittymän osa.

KUVA 2. Sisäänkirjautumisikkuna

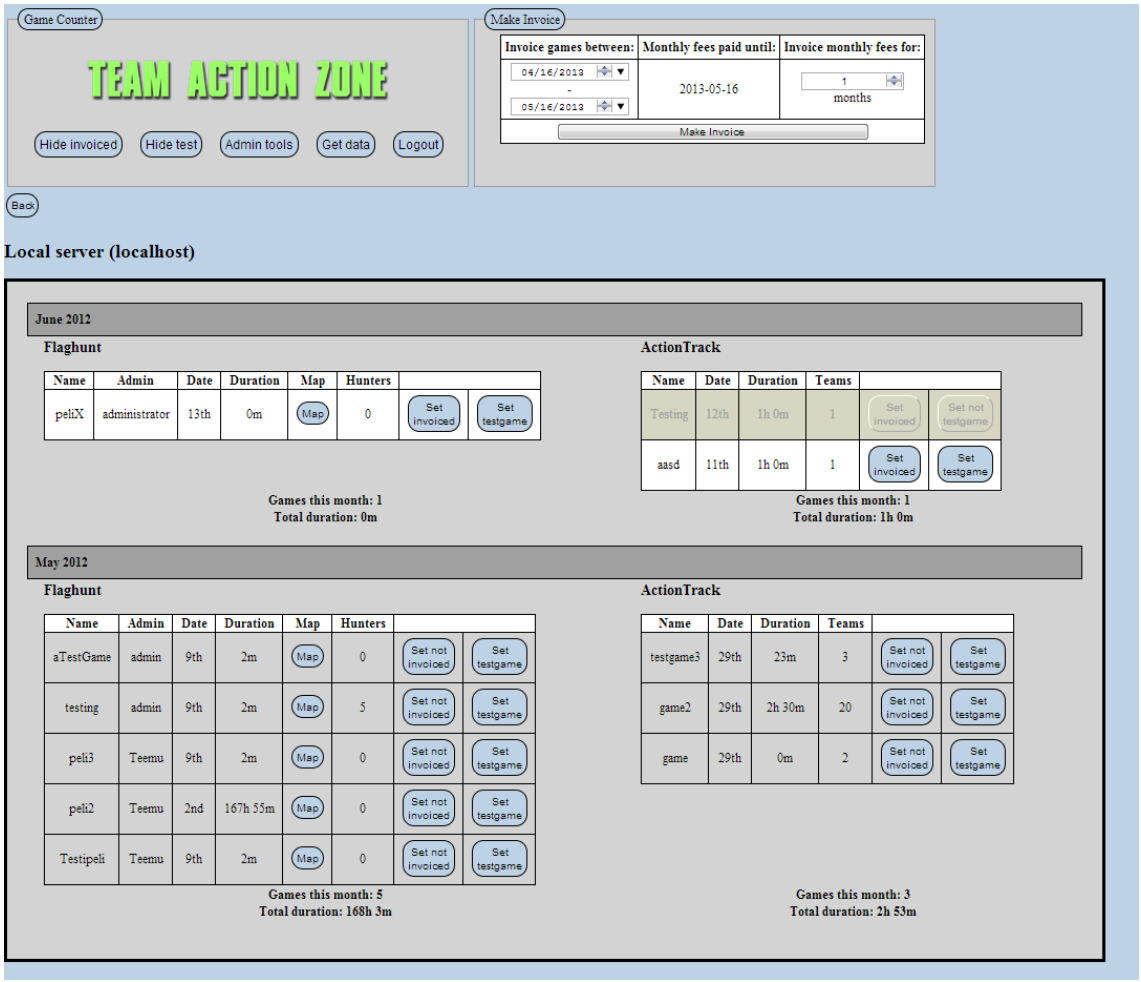
Kuvassa 2 näkyvään ikkunaan syötetään kirjautumistiedot, jonka jälkeen järjestelmään pääsee sisälle ”Login” -painikkeella.

6.3 Päänäkymä

KUVA 3. Päänäkymä

Järjestelmän päänäkymästä (Kuva 3) käyttäjä valitsee seuraavan toimenpiteen. Yläpalkista löytyvistä painikkeista pääsee esimerkiksi asetuksiin ja synkronoimaan palvelimien tiedot. Alemmasta laatikosta sen sijaan löytyvät järjestelmään syötetyt palvelimet, joita painamalla pääsee tarkemmin selaamaan palvelimen tietoja.

6.4 Palvelinnäkymä



KUVA 4. Palvelinnäkymä

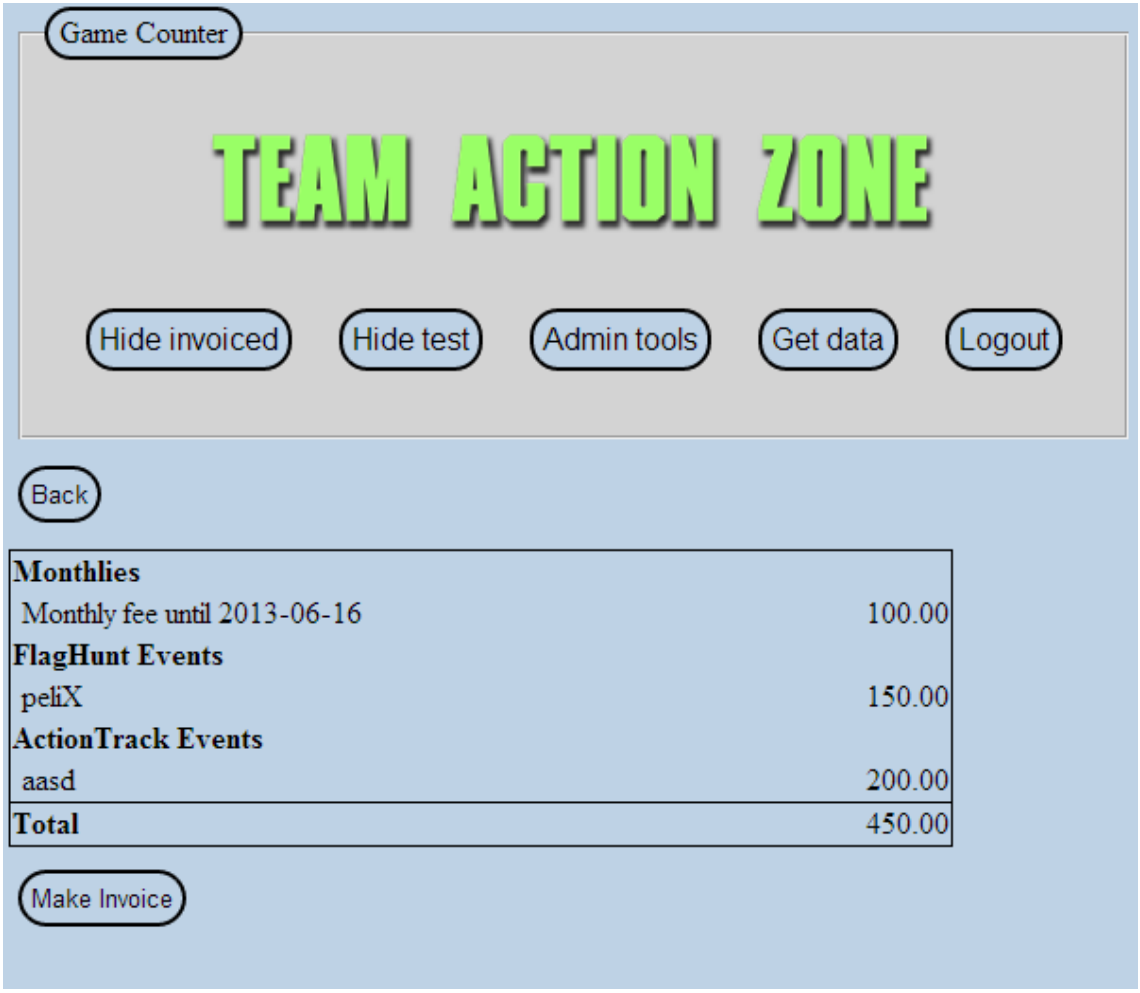
Palvelinnäkymässä on mahdollista tarkastella laskutettavia pelejä, merkata niitä jo laskutetuiksi tai laskuttamattomiksi, merkata pelejä testipeleiksi tai laskutettaviksi peleiksi, tai tehdä lasku. Näkymän yläreunassa näkyvästä laskun tekemiseen tarkoitetusta ruudusta on mahdollista määritellä miltä aikaväliltä pelit halutaan laskulle, ja monenko kuukauden kuukausimaksut tulisi laskuun lisätä.

FlagHunt pelien osalta on mahdollista myös tarkastella sijaintia, jossa peli pelattiin painamalla ”Map” -painiketta. Tämä avaa Google Mapsin uudelle välilehdelle, johon on karttaan merkitty pelin sijainti.

Molempien taulukoiden alareunassa on yhteenveto jokaisen kuukauden kohdalla kyseisen kuukauden pelatuista peleistä. Yhteenvedosta näkyy pelattujen pelien määrä ja yhteenlaskettu kesto. Testipelejä ei lasketa yhteenvetoon.

Yläreunan ”Back” -painikkeesta on mahdollista palata palvelimen valintaan.

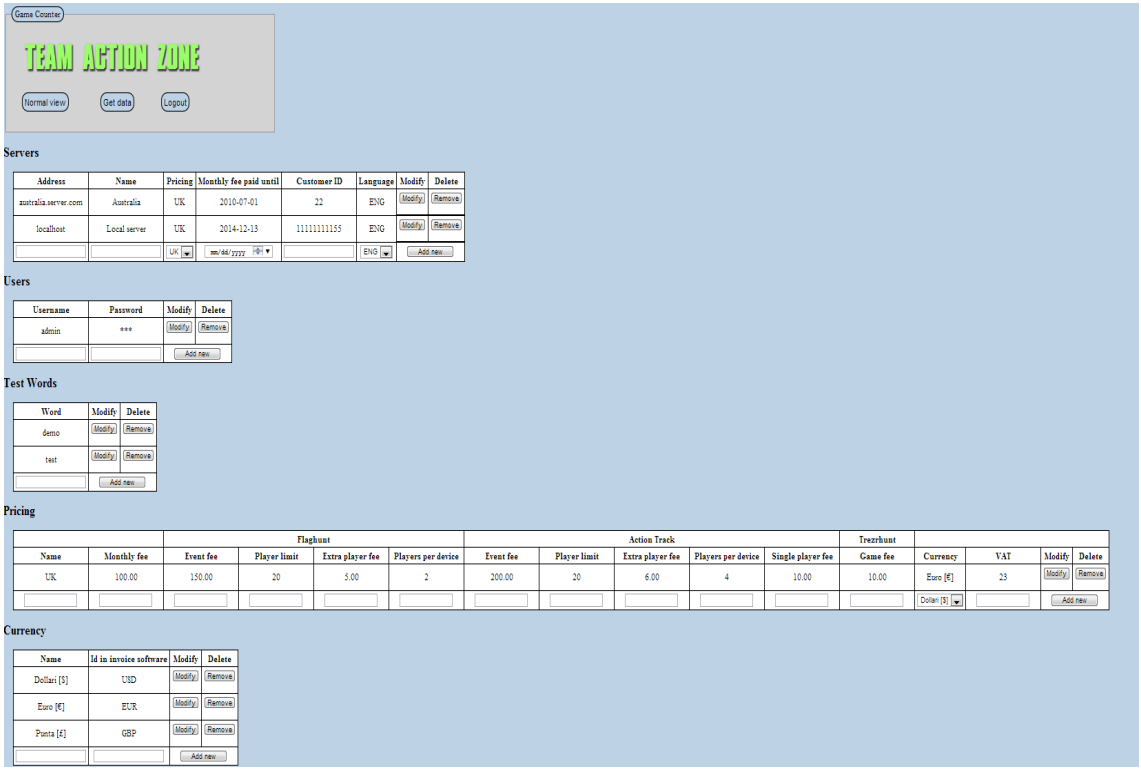
6.5 Yhteenveto



KUVA 5. Yhteenveto

Yhteenvetonäkymä näytetään käyttäjälle hänen valittuaan palvelinnäkymästä laskutettavien pelien aikaväli, ja laskutettavien kuukausien määrä. Käyttäjällä on tässä vaiheessa vielä mahdollisuus peruuttaa laskun tekeminen, tai mikäli yhteenvedossa olevat tiedot näyttävät oikeilta, hän voi tehdä näistä laskun. Lasku latautuu käyttäjän tietokoneelle .CSV muodossa käyttäjän painettua ”Make Invoice” -nappia. Tämän jälkeen kyseiset pelit merkitään tietokantaan laskutetuiksi automaattisesti.

6.6 Asetukset



KUVA 6. Asetukset

Kuvassa 6 ollaan järjestelmän asetuksissa. Tästä näkymästä käsin on mahdollista hallinnoida järjestelmän toimintaa sekä järjestelmän käyttäjiä.

Palvelimia lisättäessä tulee tietää laskutusjärjestelmästä löytyvä asiakastunniste, jota käytetään tietojen viemiseksi laskutusjärjestelmään. Palvelimen lisäyksen jälkeen palvelin ilmestyy päänäkymässä näkyvään listaan, mutta tiedot sille tulee vielä hakea käsin painamalla ”Get Data” -painiketta.

Testisanoja käytetään tietojen haun yhteydessä automaattisesti tunnistamaan mahdolliset testipelit palvelimen pelien seasta. Pelit, joiden nimestä löytyy joku testisanoista, merkitään testipeleiksi. Testipelejä voi myöhemmin muuttaa laskutettaviksi, mikäli huomataan, että järjestelmä on merkinnyt virheellisesti jonkun peleistä testipeliksi, tai toisinpäin. Näin voi käydä esimerkiksi mikäli pelin nimi on ”Demon Lair” ja sana ”Demo” on asetettu testisanaksi.

Jokaiselle palvelimelle on asetettu hinnasto, jota se noudattaa laskutiedostoa luodessa. Hinnastoja pystyy muuttamaan asetukset näkymän ”Pricing” kohdassa. Muutettavia tietoja ovat muun muassa pelikohtaiset hinnat, kuukausimaksu, valuutta sekä arvonlisäveroprosentti.

Asetuksista on myös mahdollista lisätä, muokata ja poistaa eri valuuttoja. ”Currency” -taulua käytetään yhdistämään selkokielisen tunnisteen laskutusjärjestelmän tunnisteeseen. Tällä myös pienennetään käyttäjän mahdollisia virheitä hinnastojen luomisessa, sillä pudotusvalikosta voi valita ainoastaan ”Currency” -taulusta löytyviä kohteita.

7 KÄYTTÖÖNOTTO

Ohjelmiston käyttöönotto tapahtuu kopioimalla tiedostot Apachen ”www-ssl” -kansioon. Palvelimelle, jolla itse ohjelmistoa ajetaan, on omat tiedostonsa ja palvelimille, joilta tietoa kerätään, on omansa. Mikäli molemmat laitetaan samalle palvelimelle, on myös mahdollista kerätä tietoa palvelimelta, jolla ohjelmistoa ajetaan.

Palvelimella tulee olla asennettuna PHP- ja MySQL-tuet Apacheen.

8 POHDINTA

Projekti sujui ilman suurempia ongelmia, mutta mikäli tämä tulisi toteuttaa nyt uudelleen alusta aloittaen, valikoituisi toteutustavaksi todennäköisemmin Java ja Spring Framework. Springin huomattavimpia etuja ovat todella helppo käyttöönotto useilla eri laitteistoilla, hyvä käyttäjäntunnistus sekä yksikkötestaus. Springin tapauksessa palvelinalustaksi valitsisin Ubuntu -käyttöjärjestelmän ja Tomcat -palvelinohjelmiston.

Jo suunnitteluvaiheessa olisi tullut huomioida, että on mahdollista, että tulevaisuudessa pelejä on useampia, joista tiedot laskutukseen tulee kerätä. Tämän huomioiminen vasta toteutusvaiheessa aiheutti hieman ylimääräistä työtä.

Projektin tavoitteet saavutettiin mielestäni todella hyvin, sillä kaikki toiminnalliset vaatimukset täyttyivät hyvin. Mikäli järjestelmää päätetään kehittää eteenpäin, tulee ensimmäisenä todennäköisesti vastaan eri pelien lisääminen dynaamisesti järjestelmään, ja vielä siten, että palvelinkohtaisesti olisi mahdollista valita, mitä pelejä kultakin palvelimelta tarkastellaan.

LÄHTEET

Data Collection, Wikipedia. 2013. Luettu 1.5.2013.
http://en.wikipedia.org/wiki/Data_collection

Tietokanta, Wikipedia. 2013. Luettu 1.5.2013.
http://fi.wikipedia.org/wiki/Tietokanta

Powell, G. 2005. Beginning Database Design.
Indianapolis: Wiley Publishing, Inc.