Jani Häkkinen

Kommunikaatiorajapinnan toteutus pelin ja tietokannan välille

Insinööri (AMK),

tietotekniikka

Syksy 2017

KAJAANIN
AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

TIIVISTELMÄ

Tekijä(t): Häkkinen Jani

Työn nimi: Kommunikaatiorajapinnan toteutus pelin ja tietokannan välille

Tutkintonimike: Insinööri (AMK), tietotekniikka

Asiasanat: Tietokanta, Pelit, Kommunikaatiorajapinta, PHP

Team Jolly Roger on on suomalainen pelien kehitykseen keskittynyt Kajaanilainen yritys. Sen pääosaamiseen kuuluu Unity sekä HTML5, joiden avulla tehdään omiin immateriaalisiin oikeuksiin pohjautuvia pelejä, mutta sivutoimena yritys on myös mukana alihankkijana erilaisisssa tietotekniikkaprojekteissa. Tämän opinnäytetyön tarkoituksena oli suunnitella toimiva kommunikaatiorajapinta pelin ja tietokannan välille. Peli oli opinnäytetyön kirjoituksen aikaan aktiivisessa kehityksessä oleva Neonwave Riders.

Tavoitteena oli kehittää kommunikaatiorajapinta pelin ja tietokannan välille niin, että pelaajien pelisuorituksia voi tallentaa tietokantaan ja hyödyntää muiden pelaajien käytettäväksi. Oikeiden pelisuorituksien käyttäminen pelaajan vastustajina luo tunnetta yhteisöllisyydestä ja kilpailusta pelaajien kesken. Pelistä kerättävän tiedon hyödyntäminen tietoanalyysissa on myös tärkeää pelin jatkokehittämisen kannalta.

Työ tehtiin pääasiassa PHP- ja C#-kielillä. Työn lopputuloksena oli Neonwave Riders –pelille tehty kommunikaatiorajapinta, jota voidaan hyödyntää pienillä muutoksilla myös muissa tulevissa peliprojekteissa.

ABSTRACT

Author(s): Häkkinen Jani

Title of the Publication: Communication Interface between Game and Database

Degree Title: Bachelor of Engineering, Information Technology

Keywords: Database, Games, Communication interface, PHP

Team Jolly Roger is a Finnish game development studio located in Kajaani. Studio's main focuses are Unity and HTML5, which are used to make games based on their own Intellectual property rights, but a sideline activity for the company is also working as a subcontractor on various IT projects. The purpose of this thesis was to design a functional communication interface between the game and the database. The game was at the time of the writing of the thesis in the active development as Neonwave Riders.

The aim was to develop a communication interface between the game and the database, so that the player's performances can be stored in the database and take advantage for the use of other players. Using the right game executions as player's opponents will create a sense of community, and the feel of competition between the players. Gathering collected information for data-analysis is also important in terms of further development of the game.

Development was mainly done with PHP- and C#-languages. The final result was a communication interface for Neonwave Riders game, which can be used with small changes also in the other future game projects.

SISÄLLYS

TERMILUETTELO

- **Tietokanta** Tietokanta on kokoelma yhteenliittyvää tietoa (data). Tietokannassa halutaan säilöä, muuttaa, hakea ja jakaa tietoa turvallisesti.
- **PHP** PHP on lyhenne sanoista PHP: Hypertext Preprocessor, jota käytetään erityisesti Web-palvelinympäristöissä dynaamisten web-sivujen luonnissa.
- **SQL** Structured Query Language on standarisoitu kyselykieli, jolla tietokantaan voi tehdä kyselyjä, lisäyksiä ja muutoksia.
- **Ghostdata** Pelisuorituksen sisältävä datapaketti, joka voidaan lähettää tietokantaan ja hakea tietokannasta muiden pelaajien hyödynnettäväksi. Ghostdata sisältää pelihahmon paikka- ja rotaatiotiedot kussakin ajassa sekä käytössä olevat vaatteet ym. tiedot. Ghostdatan avulla voidaan näyttää ruudulla tietokonevastustajia, joita vastaan pelaaja laskee rataa pitkin.
- **Coroutine** Coroutinet ovat funktioita, joita voi ajaa usean kuvaruudun laskemisen aikana. Coroutine funktioilla voidaan jäädä odottamaan jonkin asian valmistumista ilman, että koko ohjelma pysähtyy. Coroutine funktion tunnistaa IEnumerator sanasta funktion nimen edessä ja coroutine funktioita kutsutaan muualta koodista käskyllä StartCoroutine(funktion nimi);

1 JOHDANTO

Opinnäytetyö saatiin tilauksena Team Jolly Rogerilta, ja työ suoritettiin yrityksen tiloissa. Team Jolly Roger on kajaanilainen pelien kehitykseen keskittynyt yritys. Yritys tarvitsi toimivaa kommunikaatiorajapintaa NeonWave Riders -pelin ja SQL-tietokannan välille. Tavoitteena oli saada pelaajien pelisuoritukset tallennettua palvelimen tietokantaan, josta muut pelaajat voivat hakea aiempiin pelisuorituksiin pohjautuvia tietokonevastustajia. Työn suorittaminen edellytti aluksi perehtymistä PHP-kielen dokumentaatioon, ja myös tutoriaalivideoista oli paljon apua työn alkuun saattamiseksi. Lisäksi työssä käytettiin avuksi Unity-, XAMPP-, Apache-, WinSCP-, PuTTY- ja TortoiseSVN-ohjelmia, joihin piti myös perehtyä pintatasolla.

Kommunikaatiorajapinta toteutettiin niin, että peli voi itsenäisesti lähettää ja hakea tietoa palvelimelta lomakepyyntöjen avulla. Lomakepyynnössä määritellään, mitä tietoja halutaan tallentaa tai hakea ja kommunikaatiorajapinta antaa käskyjä näiden tietojen pohjalta palvelimen SQL-tietokannalle ja lähettää saadut tiedot edelleen pelille. Peli ei siis suoraan käskytä palvelimen tietokantaa, vaan välissä on PHP-kielellä toteutettu, suojattu palvelimella sijaitseva kommunikaatiorajapinta. Tällä tavoin varmistetaan, ettei tietokannalle vahingollisia käskyjä, kuten esimerkiksi kaiken tiedon tuhoamista, voi suorittaa manipuloimalla pelin koodia tai yhteyttä häiritsemällä.

Opinnäytetyötä varten tehtiin HTML-kielellä nettisivut, joille voitiin PHP-kielen käskyjen avulla tulostaa pelin tietokannan tietoa listattuna tekstiksi. Tietokannan tietojen listausta voidaan käyttää apuna pelinkehityksessä, pelin moderoinnissa ja lisätiedon näyttämisessä pelaajille. Tietokannan tietoa voidaan myös hyödyntää yhteisöllisyyden luonnissa, jos pelaajan kaverit tai muut pelaajat pääsevät näkemään pelisuorituksissa tallennettua tietoa.

2 TIETOKANNAN SUUNNITTELU

Tietokanta voidaan määritellä muutamalla tavalla. Tietokanta on kokoelma yhteenliittyvää tietoa eli dataa. Se on loogisesti yhtenäinen kokoelma tietoa, jolla on jokin merkitys. Tietokanta on suunniteltu, rakennettu ja täytetty tiedolla jotain tiettyä tarkoitusta varten, jolla on oma käyttäjäryhmänsä ja ennaltamääriteltyjä ohjelmia, joiden avulla käyttäjät tietokantaa hyödyntävät.

Perusvaatimuksia tietokannalle ovat tietojen säilöminen turvallisesti, tiedon helppo muokattavuus, tiedon hakeminen tietyin rajatuin ehdoin, tietokannasta yhteenvetojen ja laskutoimituksien tekeminen sekä tiedon jakaminen suurelle joukolle. Tietokanta pitäisi myös olla helposti varmuuskopioitavissa tiedon menettämisen ehkäisemiseksi.

Tietokannassa olevan tiedon toistuvuutta tulisi välttää, jotta tiedon oikeellisuus päivitettäessä säilyisi, kun tietoa ei tarvitse muuttaa monesta eri paikasta. Jos samaa tietoa tarvitaan monessa eri paikassa, tieto tallennetaan yhteen paikkaan ja siihen voidaan viitata muualta tunnistenumeron(ID) avulla. Tietoja pitää pystyä hakemaan myös sellaisin perustein, joita tietokantaa suunnitellessa ei ole pystytty ennakoimaan. Tietokannan rakennetta pitää pystyä muuttamaan jälkikäteen joustavasti tai ainakin laajentamaan ilman, että tietokantaa hyödyntävät järjestelmät eivät mene rikki. Yleensä kuitenkin joudutaan samalla myös päivittämään tietokannasta riippuvaiset ohjelmistot. [1.]

Tietokannan koko, nopeus ja käyttäjämäärän laajuuden tukeminen riippuu enemmän käytettävästä laitteistosta ja yhteyksistä kuin käytettävästä ohjelmistosta. Suurten yritysten tietokantoihin tarvitaan siis kalliimpaa laitteistoa kuin vähän käyttäjiä palveleviin pienyritysten tietokantoihin. Tietokannan suunnittelussa tulee ottaa siis ottaa huomioon tarpeen mukainen panostus laitteistoon ja siihen, kannattaako tietokantaa pyörittävä palvelimisto vuokrata vai rakentaa itse.

Vaikka tietokantaa voi reaaliaikaisesti päivittää ja tietoa hakea, joskus tietokanta voi olla niin suuri ja hakujen tarve jatkuva, että tiedon hakeminen ja listaaminen jossain tietyssä järjestyksessä, esimerkiksi jatkuvasti vaihtuvien arvojen mukaisesti, ei ole mahdollista vaan tiedot järjestetään ennalta määritellyn ajan välein ja haettaessa tätä tietoa annetaan käyttäjälle tuorein tietojen järjestetty yhteenveto. Saatujen tietojen järjestäminen voidaan jättää myös käyttäjän käyttämän ohjelman ja laitteiston tehtäväksi, jolloin resursseja säästetään palvelimilta.

2.1 Tietokannan riippuvuussuhteet

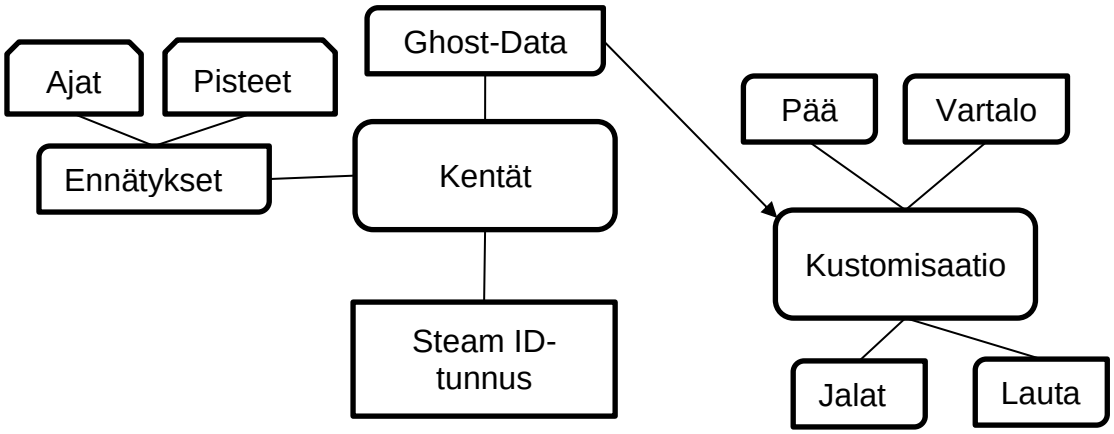
Tietokannassa oleva tieto jaetaan eri tauluihin (engl. table), jotka koostuvat tietueista (engl. record). Tietue on taulun yksi rivi joka sisältää useita kenttiä (engl. field). Kenttä sisältää tallennettavan tiedon ja kenttien nimiä voidaan ajatella taulun sarakeotsikkoina. Eri taulujen väliset tiedot voidaan yhdistää toisiinsa tunnistenumeroiden avulla, jotka ovat samanarvoisia eri tauluissa. Tämä mahdollistaa erilaisia tietojen riippuvuussuhteita.

Riippuvuussuhde voi olla yhden suhde yhteen (1-to-1), jolloin yhden taulun tietueen tieto voi liittyä vain toisen taulun yhteen tietueen tietoon: Esimerkiksi puhelinnumeron x voi omistaa vain yksi henkilö y. Riippuvuussuhteella yhden suhde moneen (1-to-M) yhden taulun tietueen tieto voi liittyä toisen taulun moneen eri tietueen tietoon. Esimerkiksi yksi henkilö y voi omistaa monia puhelinnumeroita z, x ja q. Riippuvuussuhteella monen suhde moneen (M-to-M) yhden taulun monen tietueen tieto voi liittyä toisen taulun moneen tietueen tietoon. Esimerkiksi henkilö voi omistaa monia asuntoja, mutta jonkin asunnon omistajuus voi olla jaettu usean henkilön kesken.

Tietokantaa suunnitellessa on otettava huomioon, millaisia riippuvuussuhteita eri tiedoilla on ja sen mukaan rakennettava taulut ja niiden rakennesisältö ja tauluja yhdistävät tunnistenumerot. Monen suhde moneen -riippuvuussuhteessa on luotava kahden eri taulun väliin erillinen taulu, jossa listataan tunnistenumeroiden avulla kaikki erilaiset tietojen riippuvuussuhteet. [2.]

2.2 Tietokannan rakennekaavio

Tietokannassa pelaajat erotellaan toisistaan muuttumattoman ja yksilöllisen Steam ID-tunnuksen avulla. Steam ID-tunnusta käytetään myös sitomaan tietokannan eri taulut yhteen. Näin on helppo hakea, mikä on tietyn pelaajan paras aika ja pistemäärä tietyssä kentässä tai mitä vaatteita pelaajalla on päällä. Jokaisen radan läpäisyn jälkeen tallennetaan pelaajan suoritus (Ghostdata) kyseiseltä radalta. Ghostdataan talletetaan myös, mitä vaatteita pelaajalla oli radan laskemisen aikana päällä. Kuvassa 1 on alustavien suunnitelmien mukainen tietokannan rakenne. Koska itse peli oli opinnäytetyön valmistumisen aikaan keskeneräinen, voi tietokannan rakenne vielä muuttua huomattavasti valmiiseen peliin.



Kuva 1. Tietokannan alustava rakenne

3 TYÖKALUJEN ESITTELY

3.1 Unity

Unity on Unity Technologiesin kehittämä monialustainen pelimoottori, joka tukee kaikkia merkittäviä konsoli- ja mobiilialustoja, kuten Playstation 4, XBOX ONE, Wii U, IOS ja Android. Sillä voi tehdä pelejä myös PC-, Linux- ja Mac-alustoille sekä nettisivuille pyöritettäväksi. Uusin aluevaltaus on eri virtuaalitodellisuuslasien ja -teknologioiden tukeminen. Unity-pelimoottoria on kehitetty vuodesta 2005 alkaen. Tämän opinnäytetyön projektissa käytettiin uusinta Unity 5-versiota. [3.]

3.2 XAMPP

XAMPP on lyhenne seuraavien sanojen alkukirjaimista: X (eli cross-platform), Apache HTTP Server, MySQL (tai MariaDB), PHP ja Perl. XAMPP-ohjelmistopaketin avulla kehityksessä olevia verkkosivuja voidaan näyttää omalta tietokoneelta ilman internetiä. XAMPP-ohjelmapakettia kehittää Apache Friends. [4.]

3.3 Apache

Apache HTTP Server on maailman suosituin HTTP-palvelinohjelma, jota kehittää Apache Software Foundation. Tutkimuksen mukaan 46,41 % kaikista nettisivuista arvioitiin käyttävän Apachea vuonna 2016. Apache tukee kaikkia moderneja käyttöjärjestelmiä mukaan lukien Linux ja Windows. [5.]

3.4 MySQL

MySQL on suosituin avoimeen lähdekoodin perustuva tietokantaohjelmisto. Sen merkittävimpiin käyttäjiin lukeutuvat muun muassa Google, YouTube, Yahoo, NASA ja Facebook. Ohjelmistoa kehittää Oracle Corporation. MySQL-tietokannan loi vuonna 1995 suomalainen Michael “Monty” Widenius sekä ruotsalainen Davis Axmark. MySQL-tietokantaohjelmisto myytiin Sun Microsystemsille, josta omistus siirtyi myöhemmin Oracle Corporationille. Michael Widenius on myös kehittänyt kilpailevan MariaDB-tietokantaohjelmiston, joka on hyvin samanlainen MySQL-tietokantaohjelmiston kanssa. [6.] [7.]

MySQL-palvelinta voidaan ajaa pöytätietokoneelta, kannettavalta tietokoneelta, tarkoitukseen pyhitetyllä palvelinkoneella, vuokratulta pilvipalvelimelta tai suurilta pilvipalveluklustereilta riippuen käyttäjän tarpeista ja asiakkaiden (Client) määrästä. MySQL tukee monisäikeistä ajoa. [7.]

3.5 PHP

PHP on nettisivujen toteutukseen tarkoitettu ohjelmointikieli, jota suoritetaan palvelimella ennen tuloksen lähettämistä nettisivulle, jonka ansiosta PHP ei vaadi tukea selaimelta. PHP:n avulla voi käsitellä esimerkiksi palvelimen tiedostoja ja tietokantoja, minkä vuoksi se soveltui myös hyvin opinnäytetyön kommunikaatiorajapinnan tekemiseen. PHP:n käyttäminen vaatii palvelimen, mutta koodia voi testata myös omalla koneella XAMPP-ohjelman avulla. PHP-koodia voidaan upottaa HTML-sivun sisään merkintöjen <?php ja ?> väliin ja käyttämällä tiedostopäätettä .php päätteen .html sijasta. [8.]

3.6 phpMyAdmin

phpMyAdmin on internetselaimen avulla käytettävä MySQL- ja MariaDB-tietokannan hallintatyökalu. Ohjelman kehittämisen aloitti Tobias Ratschiller vuonna 1998, ja vuonna 2001 kehitys siirtyi Olivier Müllerin, Marc Delislen sekä Loïc Chapeauxin vastuulle. [9.]

Graafisen käyttöliittymän avulla voi tehdä lukuisia tehtäviä korvaten komentorivin käytön, kuten esimerkiksi tietokantojen, taulujen, tietueiden sekä kenttien luonnin, muokkauksen ja poistamisen, tietotyyppien valitsemisen sekä käyttäjäoikeuksien hallinnoinnin. [9.]

Ohjelman avulla on helppo luoda projektin tarpeeseen vastaava tietokantarakenne, jolloin kommunikaatiorajapinnan tehtäväksi muodostuu ainoastaan tiedon välittäminen palvelimen tietokannan ja pelin välille, eli tietokannassa olevan sisällön ajankohtaisuuden ja oikeellisuuden ylläpitäminen. [9.]

3.7 PuTTY

PuTTY on telnet- ja ssh-asiakasohjelma, joka ei vaadi asennusta toimiakseen. Pääteohjelma voidaan käynnistää nopeasti lataamalla tai siirtämällä exe-tiedosto koneelle, mikä nopeuttaa ja helpottaa ohjelman käyttöä. PuTTYn avulla voidaan hallinnoida palvelinta etänä. PuTTYa tarvitaan esimerkksi Linux-käyttöjärjestelmän, phpMyAdmin-ohjelman ja SQL-tietokannan asentamiseen palvelimelle. [10.]

3.8 WinSCP

WinSCP on ilmainen avoimen lähdekoodin SFTP-, FTP-, WebDAV- ja SCP-tiedostonsiirtoprotokollia tukeva turvallisen tiedostonsiirron mahdollistava ohjelma Windowsille. Ohjelman työstämisen aloitti Martin Prikryl vuonna 2000, ja

ohjelmaan julkaistaan uusia päivityksiä edelleen. Tässä projektissa WinSCP-ohjelmaa käytettiin PHP-tiedostojen siirtämiseen tietokoneelta palvelimelle. [11.]

4 TOTEUTUS

4.1 Pelin esittely

Neonwave Riders on futuristinen kolmannesta persoonasta kuvattu lautailupeli, jossa lautailija laskee kourussa, joka mutkittelee jopa fysiikan lakien vastaisesti. Peli on ulkoasultaan pelkistetty, jossa lähes kaikki asiat maailmassa loistavat neonvalon tavoin. Pelin tavoite on suorittaa radat mahdollisimman nopeasti sekä suorittaa temppuja, joista saa pisteitä sekä kerätä erilaisia tavaroita. Peli hyödyntää tietokantaa tallentamaan pelisuorituksia sekä jakamaan niitä muille pelaajille, jotta he voivat kisata talletettuja kisasuorituksia vastaan. Tietokantaan myös tallennetaan pelaajan käytössä olevat vaatteet ja tavarat sekä statistiikkatietoa.

4.2 Steam ID

Steam ID on Steam-palvelussa myytävissä peleissä käytettävä pelaajan tilin tunnistamiseen luotu tunnistejärjestelmä. Steam ID mahdollistaa pelajan nimen ja tunnuksen pysymisen samana pelistä riippumatta sekä kaverilistat, joiden avulla pelaajat voivat kutsua kavereitansa liittymään johonkin tiettyyn peliin, sekä lukuisia muita yhteisöllisyyttä ja yksilön tunnistettavuutta parantavia ominaisuuksia.

Neonwave Riders hyödyntää näitä ominaisuuksia sekä käyttää Steam ID:n pelaajan yksilöllistä tunnusta tietokannan apuna. Koska jokaisella pelaajalla on yksilöllinen vaihtumaton Steam ID-tunnus, voi tietokantaan syöttää pelaajien tietoa, joka on sidottu Steam ID-tunnukseen. Pelissä itsessään ei myöskään tarvitse olla mitään sisäänkirjautumista, vaan peli hakee automaattsesti pelaajan Steam ID:n ja käyttää sitä tietojen hakemiseen tietokannasta.

4.3 Unity WWW-luokka

Vastuullani oli serverillä sijaitsevan PHP-rajapinnan sekä tietokannan lisäksi pelin puolella olevan yhteysrajapinnan luominen ja ylläpitäminen. Tein peliin OnlineDataManager-luokan, jonka sisällä olevia palvelimen kanssa yhteydenpitoon tehtyjä funktioita kutsutaan muualta pelin koodista. Funktioissa hyödynsin Unityssä valmiina olevaa WWW-luokkaa, joka mahdollistaa tiedon lähettämisen palvelimen rajapinnalle lomakemuodossa. Lomakkeeseen voitiin kirjata tarvittavat käskyt rajapinnalle sekä tarvittavat tiedot käskyjen suorittamista varten. Lomake voitiin lähettää yhdellä kertaa palvelimen rajapinnalle ja coroutine-tekniikan avulla funktio pystyi odottamaan palvelimen vastausta ilman, että koko peli pysähtyy täksi ajaksi.

Käskyt voivat olla esimerkiksi: tallenna seuraavat tiedot tietokantaan, hae tietoa jonkin tietyn tunnisteen, kuten pelaajan steamID-tunnisteen avulla tai hae nopeimmat ajat johonkin tiettyyn rataan. Tarvittavia tietoja voi olla esimerkiksi: salasana rajapinnan palvelimelle, steamID-tunniste, pelaajan nimi, radan nimi, radan suoritukseen käytetty aika, radassa kerätyt pisteet ja pelisuorituksen tallentava ghostdata.

Kuvassa 2 on pelin OnlineDataManager-luokan yksi coroutinefunktio, jota käytetään hakemaan radan nopeimpien pelaajien nimi-, suoritusaika- sekä pistetietoja nopeimpien aikojen perusteella. Lomakkeeseen kirjataan radan nimi, kommunikaatiorajapinnan salasana sekä nopeimpien aikojen hakemiseen tarvittava käsky, jonka jälkeen lomake lähetetään kommunikaatiorajapinnan url-osoitteeseen. Palvelimella oleva kommunikaatiorajapinta käsittelee lomakkeen, lähettää käskyn edelleen tietokannalle, vastaanottaa tulokset tietokannalta ja lähettää ne takaisin pelin OnlineDatamanager-luokan coroutinefunktiolle. Coroutinefunktio odottaa niin kauan, että kaikki tulokset ovat saapuneet

palvelimen kommunikaatiorajapinnalta ja sen jälkeen kutsuu ParseStringToList funktiota, joka muuttaa tulokset pelin ymmärtämään listapohjaiseen muotoon.

```
public IEnumerator GetHighScoreRoutine(){
                currentlevelname = UnityEngine.SceneManagement
.SceneManager.GetActiveScene ().name;
                WWWForm form = new WWWForm ();
                form.AddField ("interfacePassword", onlineinte
rfacePassword);
                form.AddField ("requestGetHighScore", 1);
                form.AddField ("levelname", currentlevelname);


                WWW www = new WWW (onlineUrl, form);
                yield return www;
                ParseStringToList (www.text);
}
```

Kuva 2. Esimerkki yhdestä funktiosta, jossa funktion ja muuttujien nimet muutettu alkuperäisestä tietoturvan vuoksi.

4.4 Palvelimen käyttöönotto

Palvelimen tarjoajaksi valitsimme Amazon Web Services-pilvilaskenta-alustan sen laajan tunnettavuuden sekä edullisen hinnan vuoksi. Palvelimia sisältäviä konesaleja on sijoitettu ympäri maailmaa, joten valitsimme lähimmän, joka sijaitsee Irlannissa. Fyysisesti mahdollisimman lähimmän konesalin valitseminen vähentää verkkoviivettä, mutta pelin julkaistaessa yksittäinen loppukäyttäjä voi olla mistäpäin maailmaa tahansa. Irlannin keskeinen sijainti tyypillisille asiakkaille on kuitenkin hyvä kompromissi.

Pelin tietokantaa varten loimme uuden instanssin(virtuaalipalvelin), johon valitsimme yhden gigatavun ram-muistia, yhden gigahertsin suorittimen ja 5 gigatavua tallennustilaa. Jos myöhemmin on tarvetta, niin suorituskykyä voi kasvattaa isommaksi tietokannan säilyessä samana. Kun instanssi oli luotu, sille määriteltiin turvallisuusasetuksia eli lähinnä mistä osoitteista ja millä tekniikalla

palvelimen eri osa-alueisiin voi ulkopuolelta olla yhteydessä. Turvallisuutta lisäten määriteltiin, että tietokantaan ei voi ottaa suoraan yhteyttä ulkopuolelta, vaikka tietäisikin tietokannan käyttäjätunnuksen ja salasanan. Ainoastaan rajapinta voi antaa valmiiksi määriteltyjä käskyjä tietokannalle ja ulkopuolelta voi ottaa yhteyden rajapintaan. Tällä pyrittiin varmistamaan, ettei tietokannalle voisi antaa käskyjä joissa esimerkiksi pyydettäisiin tuhoamaan koko tietokanta.

Instanssin luonnin ja turvallisuusasetuksien määrityksien jälkeen tarvitsi virtuaalipalvelimelle asentaa Linux-käyttöjärjestelmä sekä muita vaadittavia ohjelmia, kuten Apache, phpMyAdmin, PHP ja MySQL. Komennot ohjelmien palvelimelle asentamista varten annettiin tekstipohjaisen PuTTY-ohjelman avulla.

4.5 PHP-rajapinta

PHP-kieli ei ollut minulle ennestään tutu, joten aloitin PHP-rajapinnan koodausprojektin lukemalla PHP-dokumentaatioita sekä katsomalla paljon videotutoriaaleja aiheesta. Tähän vaiheeseen kului yllättävän paljon aikaa, koska ongelmatilanteiden sattuessa saattoi aikaa mennä tuntien verran selvittäessä vastauksia ongelmaan, joka ei ollutkaan sama ongelma, johon vastausta hain. Suurimmat haasteetkin oli selvittää, miksi jokin asia ei toiminut ja sen jälkeen etsiä siihen ratkaisu. Toisaalta vääriin ongelmiin ratkaisun löytäminen auttoi ymmärtämään PHP-kieltä laajemmalla tasolla. Näiden harjoitus-PHP-projektien pohjalta aloin tekemään varsinaista opinnäytetyön PHP-rajapintaa, ottaen parhaimmaksi havaitut tekniikat mukaan ja jättämällä vanhentuneet tai projektin kannalta turhat tekniikat pois.

Rajapintaa hyödyntävä peli oli myös tässä vaiheessa vielä alkutekijöissä, joten ei olisi ollut mahdollistakaan heti tehdä rajapintaa valmiiksi, koska itse pelissä ei ollut vielä tarvittavia asioita tiedon talteen ottamista varten, eikä tietokannasta haettua tietoakaan olisi voinut vielä missään hyödyntää. Molempien projektien edetessä yhtymäkohtia alkoi kuitenkin löytyä, jolloin rajapinnan koodaus alkoi helpottua ja tietokantaan saatiin talletettua oikeata, pelissä käytettävää tietoa.

4.6 Tietoturva:

Koska palvelimella olevan tietokannan yhteysasetus määriteltiin sisäiseksi niin, että vain rajapinta pystyy ottamaan yhteyttä tietokantaan, oli määriteltävä millä tekniikalla PHP-rajapinta ottaa yhteyden tietokantaan. Vaihtoehtoja oli käytännössä kolme: MySQL, MySQLi ja PDO, joista MySQL oli vanhentunut eikä enää virallisesti tuettu. MySQLi oli saman tekniikan uudistettu versio ja PDO tekniikka taas oli yleiskäyttöinen, joka mahdollistaa yhteyden muihinkin kuin SQL-tietokantoihin. Valitsin yhteystekniikaksi PDO:n sen monipuolisen tiedon käsittelyn vuoksi. Yleiskäyttöisen luonteensa vuoksi se on myös varmasti pitkään tuettu tekniikka. Lisäksi PDO:ssa on monia tietoturvallisuutta lisääviä ominaisuuksia, kuten prepared SQL-statements eli valmistellut SQL-käskyt.

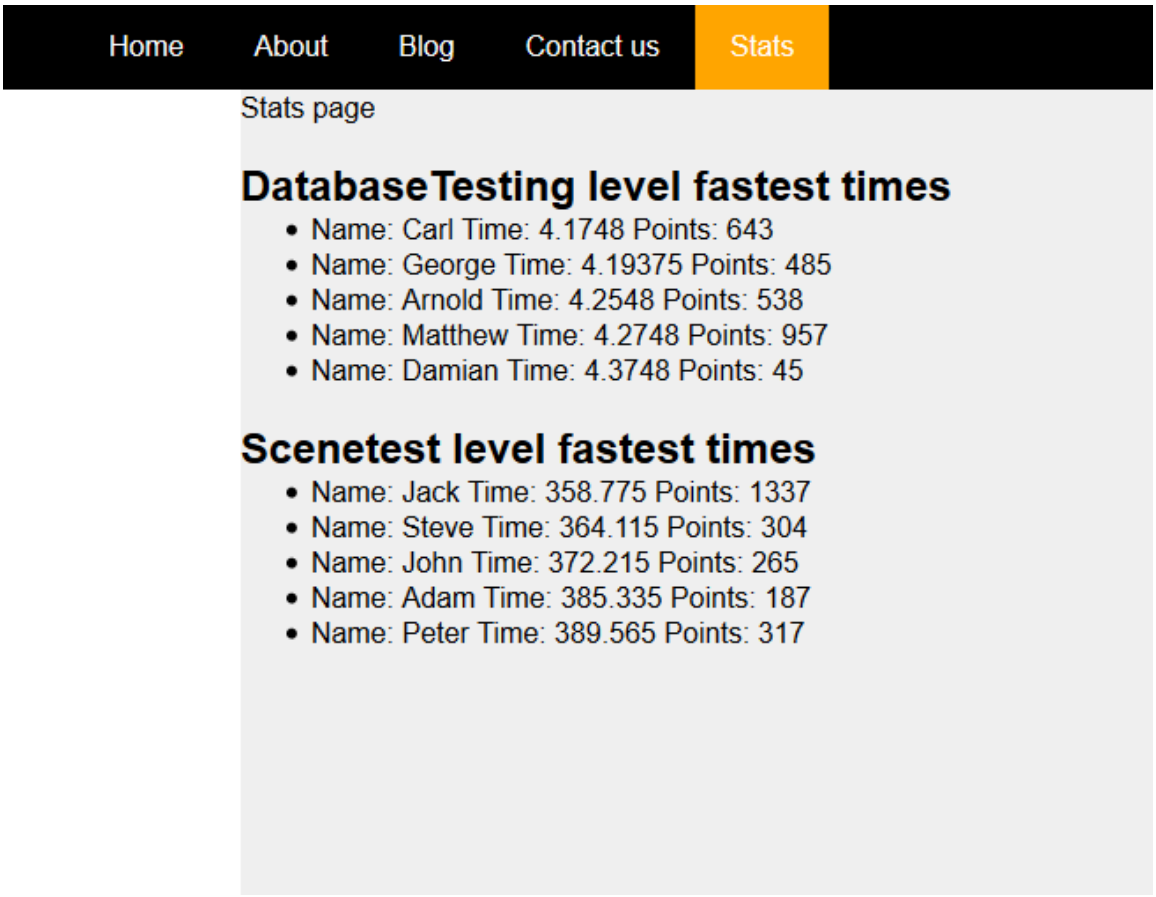
Valmistellut SQL-käskyt lisäävät tietoturvaa niin, että tietokannan käskyt määritellään ensin, jonka jälkeen määritellään annettavat tiedot. Tämä estää SQL-injektioksi kutsutun tekniikan, jossa syötetyn tiedon jälkeen onkin laitettu käsky, joka pahimmassa tapauksessa on määrätty tuhoamaan tietokanta. Kun käskyt ja tiedot määritellään erikseen, vahingolliset käskyt talletetaan tietokantaan vain tietona eikä ajeta käskynä. Valmistellut käskyt mahdollistavat myös useampien tietokokonaisuuksien lähettämisen yhdellä ja samalla valmistellulla käskyllä, eli käsky voidaan pitää samana, mutta tiedot tietokantaan lähetettävässä lausekkeessa muuttuvat.

Toinen tietoturvaa lisäävä asia on, että PHP-rajapinnalla ja SQL-tietokannalla on omat salasanansa, jotka ovat erit. Onnistuneeseen tiedon lähettämiseen tietokantaan tarvitaan siis pelin OnlineDataManager-luokka, joka lähettää lomakkeella käskyn ja tiedon lisäksi salasanan PHP-rajapinnalle jonka pitää olla sama kuin PHP-rajapintaan talletettu PHP-rajapinnan salasana. Jos salasana on oikea ottaa PHP-rajapinta yhteyden tietokantaan toisella sanasanalla, joka vastaa tietokantaan talletettua salasanaa. Jos PHP-rajapinnalle annettu salasana on väärin, ei annettuja käskyjä eikä tietoa edes käsitellä, vaan yhteys ja tiedonkäsittely lopetetaan välittömästi. Näin ei myöskään itse tietokantaa kuormiteta turhaan, koska yhteyttä tietokantaan ei tässä tapauksessa edes yritetä luoda.

4.7 Tietokannan tarkastelu www-sivuilla

Tietokannan tarkastelua varten tein yksinkertaiset nettisivut, joissa on Home-, About-, Blog-, Contact us- ja Stats-osiot. Opinnäytetyön kannalta tärkein on Stats osio joka näyttää nopeimpia aikoja pelin eri radoille. Stats-osioon on upotettu ulkopuolisille näkymätöntä PHP-koodia, joka ottaa yhteyden tietokantaan ja pyytää viiden parhaimman ajan kustakin pelin radasta. Julkisen html-koodin puolella näytetään vain PHP-koodin ajamalla saatu tulos, joka on rivejä tekstiä. Sivuilla vieraileva henkilö ei siis pääse tarkastelemaan tietokannan yhteyden saamiseen tarvittavaa koodia edes tutkimalla sivujen lähdekoodia.

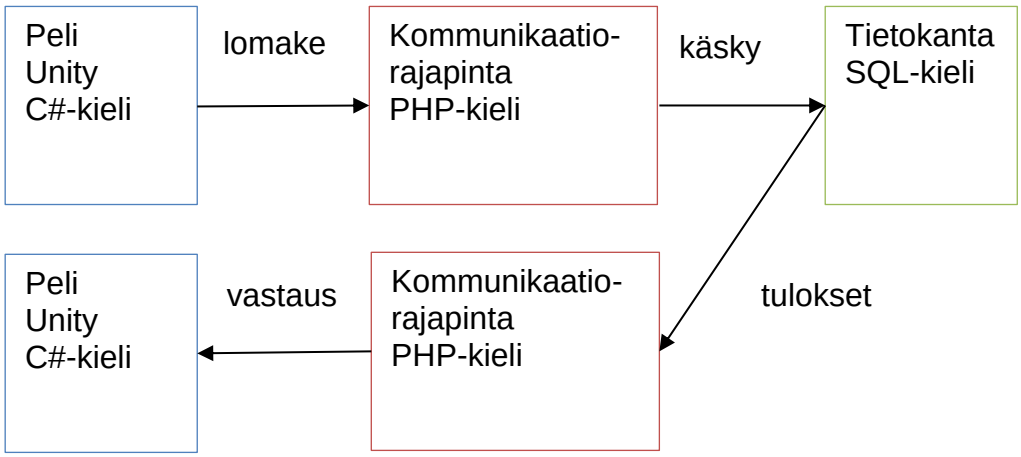
Tein myös kaikille osioille yhteisen tyyliä ja tekstinasettelua määrittelevän style.css-tiedoston sekä yhteisen header-(sivun yläreuna) ja footer-(sivun alareuna) sektiot, jotka kopioituvat jokaiseen osioon. Näin kuvassa 3 oleva sivuston tyyli pysyy yhtenäisenä ja headerissa olevat sivun navigointilinkit takaavat sivuston sujuvan selailun.



Kuva 3. Nettisivujen Stats-osiosta rajattu kuva, jossa näkyy statistiikkaa kahdesta pelin eri testiradasta.

4.8 Testaus

Testausta varten päätin mitata, kuinka kauan menee aikaa siitä, kun peli muodostaa ja lähettää lomakkeen siihen, kun vastaus on saatu palvelimelta ja käsitelty pelin tarvitsemaan muotoon kuvan 4 osoittamassa järjestyksessä. Käskynä käytin viiden nopeimman ajan hakemista tiettyyn rataan. Kommunikaatiorajapinta ja tietokanta sijaitsi palvelimella Irlannissa ja peli oli kannettavalla tietokoneella, joka oli sisäisessä wlan-verkossa Kajaanissa.



Kuva 4. Havainnekuva yhteyden kulusta pelin, kommunikaatiorajapinnan ja tietokannan välillä.

Testi tehtiin kymmenen kertaa, josta laskin viiveen eli latenssin keskiarvoksi 0,177 sekuntia pienimmän arvon ollessa 0,164 sekuntia ja suurimman arvon ollessa 0,223 sekuntia. Viiveeseen laskettiin mukaan lomakkeen muodostaminen sekä vastauksien käsittely pelin tarvitsemaan muotoon, joten testaukseen käytetyn tietokoneen suorituskyvyllä voi olla merkitystä tuloksiin käytössä olevan internet-yhteyden lisäksi.

Käyttäjämäärien kasvaessa pelin julkaisun jälkeen latenssit voivat kasvaa testausympäristössä saadusta arvosta. Arvioin kuitenkin yhden palvelimen yhdessä sijainnissa riittävän sujuvan pelikokemuksen saavuttamiseksi globaalisti, koska pelissä ei ole suurta kapasiteettia ja mahdollisimman läheistä välimatkaa vaativaa reaaliaikamoninpeliä, vaan tiedonvaihto pelin ja palvelimen välillä perustuu satunnaisiin pyyntöihin ja vastauksiin.

4.9 Työn loppuunsaattaminen

Opinnäytetyön lopuksi perehdytin yrityksen työntekijän kommunikaatiorajapinnan käyttämiseen sekä hallinnointiin ja annoin oikeudet muuttaa kommunikaatiorajapintaa, tietokantaa, pelissä olevaa OnlineDataManager-luokkaa sekä muita opinnäytetyössäni tekemiä osa-alueita vastaamaan tulevaisuuden tarpeita, kun peliprojektia kehitetään eteenpäin. Olin myös työpaikalla kirjoittamassa opinnäytetyötä työn siirron jälkeen, jotta pystyin tarvittaessa auttamaan työn jatkamisesta vastuussa olevaa työntekijää.

5 YHTEENVETO

Tämän opinnäytetyön tavoitteena oli kehittää toimiva kommunikaatiorajapinta tietokannan ja pelin välille. Tavoitteessa onnistuttiin suurimmalta osin. Opinnäytetyön alussa pelin kehitys oli alkutekijöissä, joten myös kommunikaatiorajapinnan työstäminen oli rajoitettua, koska kaikkia tarvittavia asioita ei vielä pelissä ollut olemassakaan. Opinnäytetyön edetessä myös pelin kehitysaste parani kuitenkin huomattavasti ja tietokantaan päästiin tallentamaan pelissä käytössä olevaa oikeaa tietoa.

Työ oli monipuolinen ja haastava. Se voidaan jakaa moneen osaan, jotka ajallisesti menivät osittain limittäin. Näitä ovat opinnäytetyön tavoitteen ja laajuuden määritys, PHP-kielen opettelu, työssä tarvittavien ohjelmien opettelu, palvelimen käyttöönotto ja palvelimen ohjelmien asentaminen, kommunikaatiorajapinnan teko palvelimelle, tietokannan rakenteen suunnittelu ja toteutus palvelimelle, yhteydenpitoon tarvittavan luokan teko pelin sisälle, nettisivujen teko sekä itse opinnäytetyön kirjoittaminen. Kaikista työn osa-alueista opin paljon, mutta jokaisesta osa-alueesta jäi vielä paljon syventymistä tulevaisuuden varalle.

5.1 Kehitysehdotukset

Opinnäytetyön valmistumisen aikaan pelin kehitys oli vielä kesken, joten kehityksen jatkuessa ja pelin muuttuessa myös kommunikaatiorajapintaan täytyy tehdä muutoksia. Pohjatyö kommunikaatiorajapinnassa on kuitenkin sen verran pitkällä, että näen pelinkehitystiimin pystyvän kehittämään myös kommunikaatiorajapintaa eteenpäin pelin ohella.

LÄHTEET

1. Tommi Lahtonen, Antti Ekonoja, Jukka Mäntylä & Petri Heinonen, Tietokannan suunnittelu - Luento 1. Haettu osoitteesta http://appro.mit.jyu.fi/tiedonhallinta/luennot/luento1/ (1.2.2017)

2. Advanced Kittenry – Tietokantasovellusohjeet. Haettu osoitteesta https://advancedkittenry.github.io/suunnittelu_ja_tyoymparisto/tietokanta.html (9.2.2017)

3. Unity (game engine) – Wikipedia. Haettu osoitteesta https://en.wikipedia.org/wiki/Unity_(game_engine) (13.2.2017)

4. XAMPP – Wikipedia. Haettu osoitteesta https://en.wikipedia.org/wiki/XAMPP (13.2.2017)

5. Apache HTTP Server – Wikipedia. Haettu osoitteesta https://en.wikipedia.org/wiki/Apache_HTTP_Server (13.2.2017)

6. Wikipedia – MySQL. Haettu osoitteesta https://fi.wikipedia.org/wiki/MySQL (9.2.20017)

7. MySQL :: MySQL 5.7 Reference Manual :: 1.3.1 What is MySQL? Haettu osoitteesta https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html (9.2.2017)

8. Ohjelmointiputka: Oppaat: PHP-ohjelmointi: Osa 1 – Johdanto. Haettu osoitteesta http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=php_01 (13.2.2017)

9. phpMyAdmin – Wikipedia. Haettu osoitteesta https://en.wikipedia.org/wiki/PhpMyAdmin (27.2.2017)

10. PuTTY FAQ. Haettu osoitteesta http://www.chiark.greenend.org.uk/~sgtatham/putty/faq.html#faq-what (10.4.2017)

11. Introducing WinSCP :: WinSCP. Haettu osoitteesta http://winscp.net/eng/docs/introduction (20.4.2017)