Jani Koskela

Pelikehitys CodeIgniterilla

Opinnäytetyö

Kevät 2012

Tekniikan yksikkö

Tietotekniikan koulutusohjelma

Seinäjoen ammattikorkeakoulu
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Koulutusohjelma: Tietotekniikan koulutusohjelma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Tekijä: Koskela, Jani

Työn nimi: Pelikehitys CodeIgniterilla

Ohjaaja: Mäkelä, Petteri

Vuosi: 2012 Sivumäärä: 53 Liitteiden lukumäärä: 3

Opinnäytetyössä kehitettiin Facebookissa toimiva monen pelaajan verkkoroolipeliä. Lisäksi työssä pyrittiin tutkimaan mitä hyötyjä ja haittoja on pelisovelluksen integroimisesta Facebookiin. Pelin kehitykseen pyrittiin valikoimaan hyödyllisimmät avoimen lähdekoodin työkalut ja tekniikat.

Peli ohjelmoitiin PHP:llä, JavaScriptillä ja Ajaxilla. Merkintäkielenä käytettiin HTML:ää ja tyylikielenä CSS:ää. Pelin tietokantana käytettiin MySQL-tietokantaa, jota hallinnoitiin phpMyAdminilla. Pelin kehityksessä käytettiin myös Facebook-sovellusalustan tarjoamia tekniikoita ja työkaluja. Peliin valittiin myös PHP-pohjainen sovelluskehys nimeltään CodeIgniter, mikä nopeutti pelin kehitystä huomattavasti.

Pelin kehitys saatiin pitkälle, mutta peliä ei kuitenkaan saatu julkaisukelpoiseksi. Työssä onnistuttiin kuitenkin rakentamaan kattava pelipohja, josta on helppo jatkaa pelin jatkokehitystä.

Avainsanat: PHP, CodeIgniter, Facebook, Ajax, peli, JavaScript

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Software Engineering

Author: Koskela, Jani

Title of thesis: Game Development with CodeIgniter

Supervisor: Mäkelä, Petteri

Year: 2012 Number of pages: 53 Number of appendices: 3

The purpose of this thesis was to develop a multi-player RPG (*role-playing game*) to Facebook. The study also sought out to identify the advantages and disadvantages of integrating a game application to Facebook. It was also examined which open source tools and techniques were the most optimal for the development of the game.

The game was developed with PHP, JavaScript and Ajax. HTML was used as a markup language and CSS was used as a style sheet language. MySQL-database was used to store game data and the database was controlled with phpMyAdmin. Facebook's application platform offered several techniques and tools which were used in the game development. CodeIgniter was chosen and used as a PHP-based application framework, which allowed the rapid development of the game.

The development of the game did not progress enough so that it could have been published. However, a game base was made, which offers an easy way for further development of the game.

Keywords: PHP, CodeIgniter, Facebook, Ajax, game, JavaScript

SISÄLTÖ

KUVIO- JA TAULUKKOLUETTELO

KÄYTETYT TERMIT JA LYHENTEET

PHP PHP (Hypertext Preprocessor) on HTML-dokumentteihin upotettava web-palvelimella tulkattava skriptikieli (Rantala 2005, 9).

CSS CSS on merkintäjärjestelmä, joka esittää selaimille HTML-dokumenttien ulkoasua koskevia ehdotuksia (Korpela 2008, 2).

HTML HTML määrittelee ohjeet verkkoselaimessa esitettävien web-sivujen muodostamiseen (Linjama 1998, 10).

HTTP HTTP (*Hyper Text Transfer Protocol*) on protokolla, joka mahdollistaa verkkoselaimen kommunikoinnin web-palvelimen kanssa (Parker 2005).

SQL SQL (*Structured Query Language*) on relaatiotietokantojen hallintaan tarkoitettu kyselykieli (Feuerstein ja Pribyl 2002, 3).

JavaScript JavaScript on Netscape Communications Corporationin kehittämä ensimmäinen Web-skriptauskieli (Moncur 2000, 5).

Ajax Ajax koostuu useista tekniikoista, jotka mahdollistavat asynkronisen tiedonvälityksen selaimen ja JavaScriptin, XML:n ja palvelimen välillä (Gehtland, Galbraith, ja Almaer 2006, 5).

Cron Cron on Linux- ja Unix-järjestelmissä ajoitustoimintoja suorittava sovellus (Smith 2006, 5).

MySQL MySQL on joustava, monipuolinen ja hyvän suorituskyvyn omaava relaatiotietokanta, jota käytetään web-palveluiden taustalla (Heinisuo 2004, 34).

1 JOHDANTO

1.1 Työn tausta

Työn aihevalinta perustuu tekijän henkilökohtaiseen kiinnostukseen selainpelejä kohtaan. Inspiraationa työn aloittamiselle toimivat selainpelit, kuten *Mousehunt*, *Castle Age* ja *Travian*. Edellä mainitut pelit ovat olleet hyvin suosittuja jo vuosien ajan ja ovat siten todennäköisesti tuottaneet merkittäviä voittoja niitä valmistaville yrityksille. Motivaationa työn tekemiselle toimi tekijän haaveet oman selainpelin julkaisemisesta, ja oman selainpelejä tuottavan pienyrityksen perustamisesta.

1.2 Työn tavoite

Opinnäytetyön päätavoitteena oli kehittää ja julkaista Facebookissa toimiva monen pelaajan verkkoroolipeli nimeltään *Bunny vs Magician*. Pelaajan tarkoitus on kehittää roolihahmoaan ja taistella verenhimoisia kaneja vastaan. Projektissa tutkittiin myös mitä tekniikoita ja työkaluja olisi hyödyllisintä soveltaa pelin kehityksessä, huomioiden tekijän saatavilla olevat resurssit ja varat. Tavoitteena oli myös pelin kehityksen yhteydessä havaita mitä etuja ja haittoja on pelisovelluksen integroimisesta Facebookiin. Vaikkakin opinnäytetyö keskittyy pääosin pelin ohjelmoimiseen, tavoitteena oli myös kerätä tietoa pelin suunnittelun teoriasta.

1.3 Työn rakenne

Toisessa luvussa käsitellään CodeIgniterin niitä toimintoja ja ominaisuuksia, joiden ymmärtäminen oli oleellista peliprojektin toteuttamisessa.

Kolmannessa luvussa selitetään peliprojektin lähtökohtaa ja suunnittelua. Pelin sunnittelussa päätetään pelin päätavoite ja sen teknilliset päälinjaukset. Suunnittelussa myös tutkitaan mitä tekniikoita ja työkaluja peliprojektissa tullaan hyödyntämään.

Neljännessä luvussa aloitetaan peliprojektin toteuttaminen, eli pelipohjan rakentaminen. Pelipohjan rakentamisessa käydään lävitse CodeIgniterin asennus ja käyttöönotto. Tämän jälkeen käydään lävitse Facebook-sovellusalustan käyttöönotto. Lopuksi selitetään pelissä käytettävistä reaaliaikaisista toiminnoista ja niiden käyttöönotoista.

Viidennessä luvussa selitetään peliin tehdyistä ominaisuuksista, jotka ovat oleellisia pelin pelattavuudelle ja ymmärtämiselle.

Kuudennessa luvussa esitellään pelisivuston pelisivut, joita pelaaja voi selailla verkkoselaimellaan. Luvussa selitetään pelisivujen rakenteet, ominaisuudet ja käyttötarkoitukset. Laajempien pelisivujen ominaisuudet jaettiin alaotsikkoihin.

Viimeisissä luvuissa kerrotaan peliprojektin tulokset ja yhteenveto.

2 CODEIGNITER-SOVELLUSKEHYS

2.1 Yleistä

CodeIgniter on ohjelmistokehys, jolla voidaan kehittää web-sovelluksia käyttäen PHP:tä. CodeIgniterin tavoite on nopeuttaa web-sovellusten kehittämistä tarjoamalla yksinkertainen käyttöliittymä, perusteellinen dokumentaatio, looginen rakenteen ja valmiita kirjastoja. Kirjastot sisältävät useita web-sovelluksien kehittämisessä yleisesti tarvittavia ohjelmointikomponentteja, kuten esim. lomakkeiden validoinnin. CodeIgniter on kevyt ohjelmistokehys, jonka ydin sisältää vain muutamia pieniä kirjastoja. Tarvittavat lisäkirjastot, joita käyttäjä voi luoda myös itse, voidaan ladata käyttäjän pyynnöstä. CodeIgniter on vapaan lähdekoodin ohjelmisto, eli se on vapaasti kaikkien käytettävissä. (CodeIgniter User Guide 2010b.)

2.2 MVC-kehitysmalli

CodeIgniter perustuu MVC-kehitysmalliin (*Model View Controller*), joka erottaa sovelluksen logiikan käyttöliittymästä. MVC-mallin tarkoitus on selkeyttää koko sovelluksen rakennetta, ja vähentää tarvittavan skriptin määrää varsinaisessa web-sivussa. Yleinen MVC-malli koostuu seuraavista osista:

- malli (*model*) sisältää tietokantaan liittyvät toiminnot

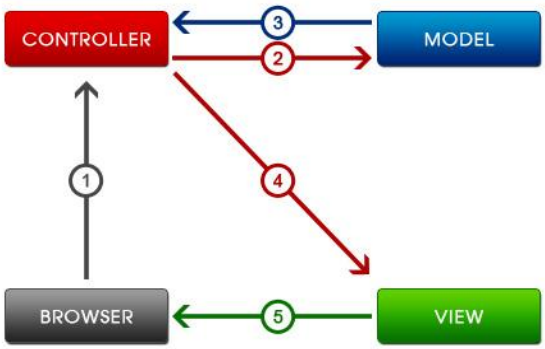
- näkymä (*view*) muodostaa käyttäjälle näytettävän datan, esimkerkiksi web-sivun tai web-sivun osan

- ohjain (*controller*) välittää tietoa näkymien, mallien ja mahdollisesti muiden HTML-pyyntöön tarvittavien resurssien välillä. (CodeIgniter User Guide 2010e.)

CodeIgniterin lähestymistapa MVC-kehitysmalliin on löysä, koska se ei vaadi mallien käyttöä. Jos esimerkiksi mallien käyttö web-sovelluksessa ei ole tarpeellista,

käyttäjä voi kehittää web-sovelluksen käyttämällä ainoastaan näkymiä ja ohjaimia. (CodeIgniter User Guide 2010e.)

MVC-mallin toimintaperiaatetta kuvataan alla olevassa kuvassa.

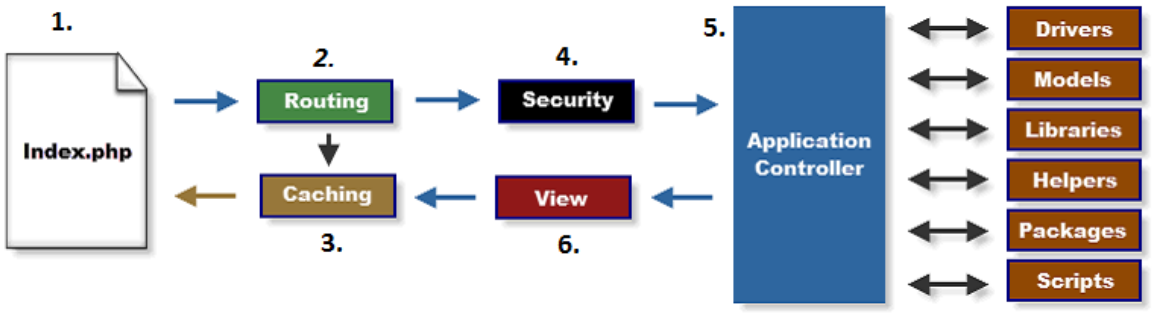


Kuva 1. MVC-mallin toimintaperiaate.

1. Selain lähettää pyynnön ohjaimeen.
2. Ohjain kommunikoi mallin kanssa.
3. Malli suorittaa prosessoinnin ja lähettää tietoa takaisin ohjaimeen.
4. Ohjain analysoi mallin lähettämät tiedot ja välittää ne näkymään.
5. Näkymä välitetään selaimeen. (Pop 2010.)

2.3 Tiedon kulku

Kuva 2 osoittaa kuinka CodeIgniter vastaa HTTP-pyyntöön.



Kuva 2. Tiedon virtaus CodeIgniterissa.

1. *Index.php*-tiedosto toimii etukontrollina, alustaen perusresurssit CodeIgniterin ajamiseen.

2. HTTP-pyyntö tarkistetaan, ja valitaan miten edetään.

3. Jos pyydetty web-sivu löytyy välimuistista, se lähetetään suoraan selaimelle, mikä nopeuttaa web-sivun muodostamista.

4. HTTP-pyyntö ja käyttäjän mahdollisesti lähettämät tiedot suodatetaan tietoturvallisuuden edistämiseksi.

5. Ohjain lataa siihen ladattavaksi määritetyt resurssit, kuten esim. auttajat, mallit tai näkymät. Ohjain myös lataa automaattisesti ladattavaksi asetetut resurssit, kuten esim. ydinkirjastot.

6. Lopulta HTTP-pyyntöön lähetetään vastaukseksi finalisoitu näkymä, jota käyttäjä voi tarkastella verkkoselaimen kautta. Muodostettu näkymä tallennetaan välimuistiin, jos välimuistiin tallentaminen on määritelty sallittavaksi. (CodeIgniter User Guide 2010a.)

2.4 Kirjastot

CodeIgniterissa kirjasto on luokka-tiedosto, joka sisältää helposti käytettäviä toimintoja. Ohjelmoija voi vähentää työmääräänsä käyttämällä kirjastoja, kuten esimerkiksi CodeIgniterin tietokanta-kirjastoa. Tietokanta-kirjastolla käyttäjä pystyy helposti luomaan monimutkaisia ja selkeitä SQL-kyselyitä. (Griffiths 2010, 29.)

CodeIgniterin kirjastot ovat vapaasti muokattavissa, käyttäjä voi laajentaa tai korvata niitä, tai luoda täysin uusia kirjastoja. Kirjastot ladataan ohjaimesta, minkä yhteydessä kirjastoon on myös mahdollista välittää parametrejä. (CodeIgniter User Guide 2012c.)

2.4.1 Relaatiotietokanta

Relaatiotietokanta jäljittelee ihmisen oikoteistä ja assosioivaa ajattelutapaa. Monimutkaisen tai vaikean kokonaisuuden ymmärtäminen helpottuu yleensä, kun se jaetaan pieniin, toisiinsa liittyviin palasiin, ja yritetään ymmärtää jokainen palanen erikseen. Relaatiotietokanta yksinkertaisesti säilöö nämä pienet palaset keskinäisine suhteineen. (Meloni 2003, 8.)

Relaatiotietokanta koostuu yhteen liitetyistä tauluista, jotka koostuvat riveistä ja sarakkeista. Taulut liitetään toisiinsa sarakkeisiin merkittyjen arvojen perusteella. (Meloni 2003, 8.)

Relaatiotietokannat ovat hyvin tärkeässä roolissa web-sovelluksissa. Ilman tietokantoja web-sovellukset eivät pystyisi varastoimaan tietoja. CodeIgniteri tarjoaa tietokanta-kirjaston, jolla pystyy helposti ja tehokkaasti kommunikoimaan tietokantojen kanssa. (Griffiths 2010, 77.)

2.4.2 Tiedostojen lataus

CodeIgniter sisältää kirjaston, jolla voi helposti ladata tiedostoja web-palvelimelle. Kirjastolla voi määritellä sallittujen tiedostojen ominaisuuksia, kuten esim. tiedoston koon tai tiedostopäätteen. (Griffiths 2010, 43-44.)

2.5 Avustajat

Avustajien (*Helpers*) tarkoitus on helpottaa ohjelmointia tarjoamalla erilaisia funktioita, joita käyttäjä voi tarvittaessa ottaa käyttöön. Avustajia voi esimerkiksi käyttää lomakkeiden prosessoimiseen tai evästeiden hallintaan. Yksittäinen avustajatie-

dosto koostuu itsenäisistä funktioista, jotka on helppo ladata. Oletuksena avustajien lataus tulee tehdä manuaalisesti, mutta avustajien lataaminen voidaan myös asettaa automaattiseksi. Avustajien lataukset on suositeltavaa suorittaa ohjaimessa, mutta ne voidaan myös ladata näkymissä. Avustajia on myös mahdollista ladata ohjaimen konstruktoriin, jolloin avustajat ovat saatavilla ohjaimen jokaisessa metodissa. Kun avustaja on ladattu, avustajan funktiot ovat käytettävissä, ja niitä voidaan kutsua kuten normaaleja PHP-funktiota. (CodeIgniter User Guide 2012d.)

2.6 XSS-suodatin

XSS (*Cross site scripting*) on tietoturva-aukko, jossa pyritään syöttämään vahingollista JavaScript-koodia sovellukseen. Kyseistä tietoturva-aukkoa voi hyödyntää esim. kaappaamalla toisen käyttäjän ID:n. (Griffiths 2010, 169.)

CodeIgniter sisältää XSS-suodattimen, joka voidaan ajaa tarvittaessa, tai se voidaan asettaa suodattamaan kaikki POST- ja evästetiedot automaattisesti. XSS-suodatinta ei ajeta oletuksena, koska sen ajamista ei aina tarvita. (Griffiths 2010, 34.)

3 PELIN SUUNNITTELU

3.1 Lähtökohta

Pelin kehityksen lähtökohtana toimi tekijän vuokraama web-hotelli. Peliprojektissa käytettävän web-palvelimen ylläpitoon käytettiin web-hotellin tarjoamaa cPanelin graafista käyttöliittymää. cPanel sisältää muunmuassa koodieditorin, hakemistoselaimen ja varmuuskopiointisovelluksen, joita käytetään runsaasti peliprojektin aikana. Web-palvelimen lisäksi web-hotelli tarjoaa MySQL- ja PostgreSQL-tietokantoja ja niiden hallintaan tarkoitettuja graafisia käyttöliittymiä.

3.2 Pelin päätavoite

Pelin suunnittelu alkaa selkeästi asetetulla päätavoitteella, esimerkiksi maailman pelastaminen avaruusolioiden invaasiolta. Päätavoite ohjaa pelin tarinan kulkua ja määrittelee onko peli innovatiivinen. (Lobâo, Evangelista, Farias & Grootjans 2009, 4.)

Pelin päätavoitteeksi ideoitiin, että pelaajan on torjuttava brutaalien kanien invaasio taistelemalla niitä vastaan. Pelaaja asettuu velhon rooliin, ja ryhtyy kehittämään roolihahmoansa paremmaksi taistelijaksi, mikä mahdollistaa voimakkaampien kanien voittamisen. Kaneja voittamalla pelaaja voi saada kokemusta, kultaa ja tarvikkeita. Kullalla pelaaja voi ostaa tarvikkeita, jotka nostavat pelaajan taistelukykyä. Kokemuksella pelaajan kokemustaso nousee, jolloin pelaaja saa valita itselleen lisäominaisuuksia. Lisäominaisuuksia voi myös saada voittamalla kaksintaisteluja toisia pelaajia vastaan. Lisäksi pelaaja voi muodostaa ryhmiä tai liittyä toisten pelaajien ryhmiin. Ryhmissä pelaajat voivat saada lisäominaisuuksia ja taistella pelimaailman voimakkaimpia kaneja vastaan. Pelaajat voivat vapaasti matkustaa pelimaailman eri kohteissa ja taistella siellä olevia kaneja vastaan.

3.3 Pelin teknillinen suunnittelu

Peli kehitettiin verkkoselaimessa pelattavaksi monen pelaajan verkkoroolipeliksi, joka käyttää Facebookin sovellusalustaa. Facebookin sovellusalusta tarjoaa pelisovellukselle erinomaiset markkinointimahdollisuudet, joita hyödynnettiin pelin kehityksessä. Peli muodostuu ainoastaan kuvista ja tekstistä, eikä se sisällä animaatioita (lukuun ottamatta kevyitä JavaScript-animaatioita). Tämän vuoksi pelin pelaamiseen ei vaadita verkkoselaimien lisäosien asennuksia.

Peli kehitettiin siten, että se tukee suosituimpien verkkoselaimien (Mozilla Firefox, Google Chrome, Opera, Internet Explorer) uusimpia versioita. Tämä saavutettiin muun muassa ohjelmoimalla yksinkertaista ja standardisoitua HTML-koodia, ja käyttämällä CSS-pohjaista sovelluskehystä nimeltään Blueprint. Blueprint-sovelluskehystä käytettiin HTML-elementtien asetuksien määrityksessä.

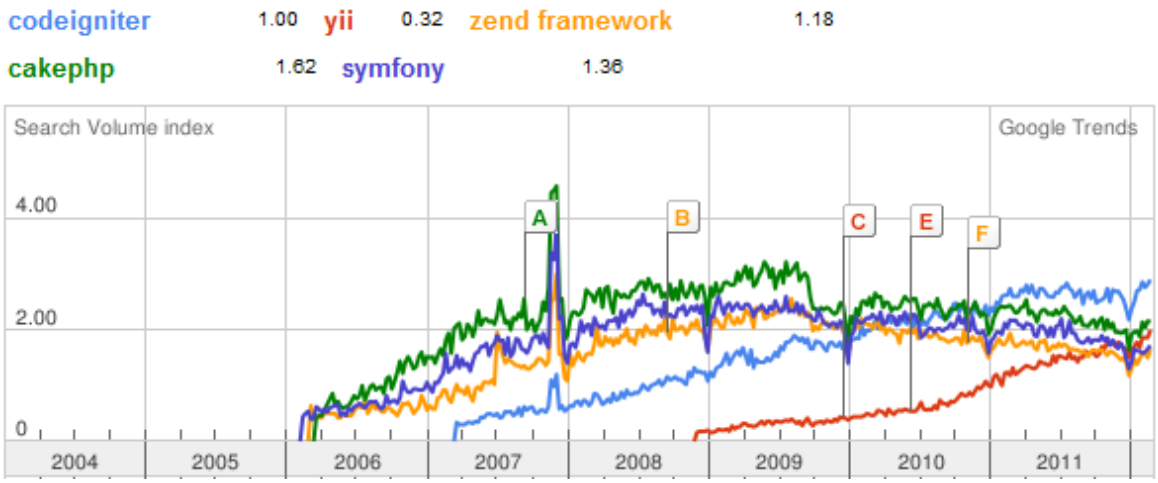
Pelin kehittämiseen valittiin palvelimella suoritettava skriptikieli. Skriptikielen tehtävänä oli suorittaa suurin osa pelin logiikasta ja kommunikoida tietokannan kanssa. Peliprojektissa käytettävä web-hotelli tarjosi useita potentiaalisia vaihtoehtoja palvelinpuolen skriptikielen valintaan, kuten muun muassa Perl, PHP ja Ruby on Rails. Peliprojektin nopeuttamiseksi palvelinpuolen kieleksi valittiin lopulta PHP, koska tekijällä oli aikaisempia kokemuksia PHP:n käytöstä.

Pelistä odotettiin tulevan varsin laaja web-sovellus, joten haluttiin että pelisovelluksen looginen rakenne olisi selkeä ja helppo ymmärtää. Laajan web-sovelluksen kehittäminen ”puhtaalta pöydältä” olisi ollut myös hyvin työläs projekti, joten pelille päätettiin etsiä PHP-pohjainen sovelluskehys. Sovelluskehykseltä etsittiin seuraavia ominaisuuksia:

- hyvä suorituskyky ja tietoturva
- Facebook-sovellusalustan yhteensopivuus
- kattava ja selkeä dokumentaatio
- yhteisön laajuus ja aktiivisuus
- avoimen lähdekoodin ohjelmisto

- yleinen suosio ja uskottavuus.

Yllä mainitut kriteerit täyttyivät tyydyttävästi useissa PHP-pohjaisissa sovelluskehyksissä, kuten esimerkiksi CodeIgniterissa, CakePHP:ssä, Yiissä, Zendissä ja Symfonyssa. Sovelluskehykseksi valittiin lopulta CodeIgniter (versio 2.1.0), sen selkeän dokumentaation ja aktiivisen yhteisön vuoksi. CodeIgniter oli Google Trendsin mukaan myös lähivuosina yksi etsityimmistä PHP-pohjaisista sovelluskehyksistä (ks. kuva 3).



Kuva 3. PHP-pohjaisten sovelluskehysten Google-hakujen vertailua.

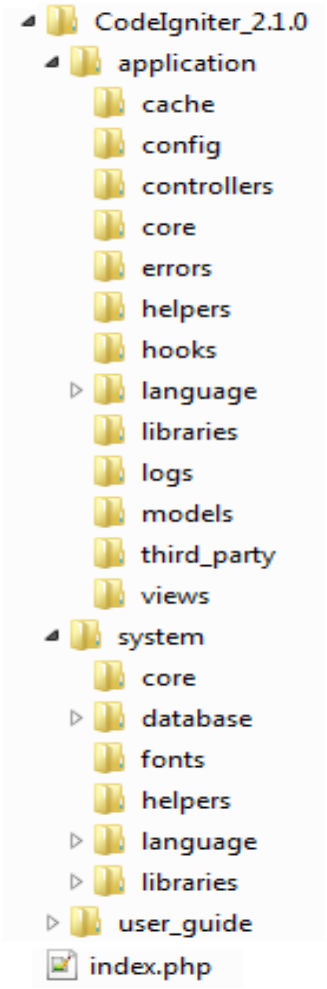
Vaikkakin kuvassa 3 näkyvien sovelluskehyksien hakumäärien erot ovat suhteellisen pieniä, se kuitenkin osoittaa että CodeIgniter on yksi yleisimmistä PHP-pohjaisista sovelluskehyksistä. Sovelluskehyksen yleisyys kasvattaa sen yhteisön aktiivisuutta ja sovelluskehyksen uskottavuutta.

Pelin yksi tärkeimmistä ominaisuuksista on kyky tallentaa pelaajan ja pelimaailman tietoja. Näiden tietojen tallennusta varten valittiin tietokanta. CodeIgniter tukee peliprojektissa käytettävän web-palvelimen molempia tietokantatyyppejä, MySQL:ää ja PostgreSQL:ää. Vaikkakin molemmat tietokannat olisivat soveltuneet peliprojektiin, pelissä käytettiin MySQL-tietokantaa, koska tekijällä oli aikaisempia kokemuksia MySQL-tietokantojen käytöstä. MySQL-tietokantaa hallinoitiin web-hotellin tarjomalla phpMyAdminin graafisella käyttöliittymällä.

4 CODEIGNITER PELIPOHJANA

4.1 Asennus ja käyttöönotto

CodeIgniter asennettiin lataamalla CodeIgniterin lähdekoodi sen virallisilta sivuilta (http://codeigniter.com), minkä jälkeen ladattu lähdekoodi-paketti (ks. kuva 4) purettiin web-palvelimelle.



Kuva 4. CodeIgniterin kansiorakenne.

Yllä olevassa näkyvät *application*- ja *system*-kansiot purettiin tietoturvan edistämiseksi web-palvelimen juureen, jotta niihin ei päästäisi verkkoselaimen kautta. Yllä olevassa kuvassa näkyvä *index.php*-tiedosto siirrettiin julkiseen kansioon, johon päästään verkkoselaimen kautta. *index.php*-tiedostoon määriteltiin sen jälkeen

application- ja *system*-kansioiden uudet sijainnit. Tämän jälkeen CodeIgniter oli käyttövalmiina. Pelin kehityksen ajaksi *index.php*-tiedostossa työympäristöksi määriteltiin *Development*, joka sallii virheilmoitusten näytön web-sivuilla. Tämä helpottaa mahdollisten koodivirheiden paikannusta.

CodeIgniterin purettu lähdekoodi sisälsi ohjaimen, jota käytetään pelin kehityksessä. Peliprojektissa ohjainta käytetään HTTP- ja Ajax-pyyntöjen vastaamiseen. Näitä pyyntöjä varten ohjaimeen luodaan metodeja, joissa määritellään miten näihin pyyntöihin vastataan. Ohjaimen metodeihin määritetään mitä resursseja niissä ladataan ja ajetaan. Ohjaimen sijainti on *application/controllers*-kansiossa.

Peliprojektissa luodut mallit luotiin *application/models*-kansioon, ja näkymät luotiin *application/views*-kansioon.

MySQL-tietokannan käyttöönottoon tarvittavat asetukset säädettiin *database.php*-tiedostossa, joka sijaitsi *application/config/*-kansiossa.

4.2 Facebook-sovellusalustan käyttöönotto

Facebook-sovellusalustan käyttöönottoa varten käytettiin Alfonso E Martinezin kehittämää, avoimen lähdekoodin Facebook Ignitedia (versio 1.0.7), joka on lisäosa CodeIgniteriin. Ennen Facebook Ignitedin asennusta luotiin Facebook-sovellus Facebook developerin sivuilla (https://developers.facebook.com/). Luodun sovelluksen jälkeen asennettiin Facebook Ignited, mikä tapahtui yksinkertaisesti korvaamalla palvelimella oleva CodeIgniterin lähdekoodi Facebook Ignitedin lähdekoodilla. Asennuksen jälkeen Facebook-sovelluksen asetukset, jotka saadaan Facebook developerin sivuilta, määriteltiin Facebook Ignitedin asetuksiin (ks. liite 1). Tämän jälkeen Facebook-sovellusalustan ominaisuudet olivat käyttövalmiina.

Peliprojektissa tärkein ja käytetyin Facebook-sovellusalustan välittämä tieto on Facebook-käyttäjän tunnistenumero, jota käytetään pelin jokaisella web-sivulla. Jokaisella sivulatauksella tunnistenumero tallennetaan CodeIgniterin asetuksiin, josta se on helppo noutaa tarvittaessa. Tunnistenumeron tallennus määritetään seuraavan koodin mukaisesti CodeIgniterin ohjaimen konstruktorissa, jolloin tunnistenumero päivitetään jokaisella sivupyynnöllä.

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Pages extends CI_Controller {

	function __construct()
	{
		parent::__construct();
		$this->fb_me = $this->fb_ignited->fb_get_me();
		$user_id = $this->fb_me['id'];
		$this->config->set_item('user_id', $user_id);
	}
}
```

4.3 Pelin reaaliaikaiset toiminnot

Pelin reaaliaikaiset ominaisuudet kehitettiin JavaScriptillä ja sen lisäkirjastolla, jQueryllä. JavaScriptin käyttöä peliprojektissa korostettiin, koska skriptit ajetaan käyttäjän selaimessa eikä palvelimella. Tämä vähentää palvelimen kuormitusta. Kun pelaaja tekee pelissä kokonaisen sivulatauksen, palvelin prosessoi sen ja lopulta vastaa siihen lähettämällä sisällön pelaajan verkkoselaimelle. Tämän vastauksen yhteydessä pyritään lähettämään myös kysytyn web-sivun mahdolliset alalinkit, joiden sisällön näkyvyyttä kontrolloidaan JavaScriptillä. Pelissä alalinkkien sisällöt näytetään yleensä yksitellen: kun pelaajalle näytetään yhden alalinkin sisältö, muiden alalinkkien sisällöt piilotetaan. Esimerkiksi jos pelaaja lajittelee inventaariostaan näkyviin vain aseet, muu inventaario piilotetaan. JavaScriptin avulla alalinkkien sisältöjen selaaminen on nopeaa, ja koska sisältöä ei haeta uudestaan palvelimelta, palvelimen kuormittuminen vähenee.

Facebookin sovellusalusta tarjoaa JavaScript-kirjaston, joka ladattiin liitteen 2 mukaisesti. Tämän jälkeen Facebook-alustan reaaliaikaiset ominaisuudet olivat käytettävissä. Latauskoodi asetettiin omaan JavaScript-tiedostoon, josta se linkitettiin niille pelin web-sivuille, joihin tarvittiin Facebook-alustan JavaScript-ominaisuuksia.

Suurin osa pelin tietokantaan tehtävistä muokkauksista suoritetaan reaaliaikaisesti Ajax-tekniikalla. Esimerkiksi jos pelaaja ostaa uuden aseen, palvelimelle lähetetään siitä tieto Ajaxin avulla, minkä jälkeen palvelin prosessoi saadun tiedon ja lähettää lopulta vastauksen takaisin pelaajan verkkoselaimeen. Ajax-kutsujen yhteydessä yleensä välitetään parametrejä, jotka poikkeuksetta validoidaan palvelinpuolella. Parametreinä voi esimerkiksi olla pelaajan ostaman aseen tunnistenume-

ro, jonka perusteella tehtäisiin muutoksia tietokantaan. Palvelinpuolella CodeIgniterin ohjain vastaanottaa Ajax-kutsut metodeissaan, ja määrittelee niiden prosessoinnin. Kun Ajax-kutsu on suoritettu, käyttäjälle yleensä ilmoitetaan oliko Ajax-kutsu onnistunut, esimerkiksi jos pelaajan aseen osto onnistui, siitä ilmoitetaan lopuksi pelaajalle. Ajax-kutsuja pyritään kuitenkin välttämään, koska niiden prosessoiminen kuormittaa palvelinta. Liitteessä 3 esitetään jQueryllä ohjelmoitu, peliprojektissa käytettävän Ajax-kutsun perusrakenne.

5 PELIOMINAISUUDET

5.1 Pelin sivupohja

Pelille kehitettiin sivupohja, jota käytetään pelin jokaisella sivulla. Sivupohjan kehittäminen aloitettiin hahmottelemalla pelin etusivun ulkoasu. Hahmotellun ulkoasun jälkeen päätetään miten pelipohja muodostetaan. Pelin ulkoasu hahmoteltiin käyttäen tekijän omistamaa Photoshop CS5 -lisenssiä. Alla olevassa kuvassa näkyy pelin sivupohjan hahmottelua.

Kuva 5. Pelipohjan hahmottelua Photoshop CS5:lla.

5.1.1 Sivupohjan yläosa

Kuvassa 5 näkyvä sivupohjan yläosa jakaantuu kolmelle riville:

- ylin rivi sisältää logon, uutisosan ja ryhmien mainostusosan
- keskimmäisessä rivissä näytetään pelaajan tilatietoja, kuten esimerkiksi pelaajan sijainti
- alimmalle riville tulee päälinkit, joiden avulla käyttäjä voi selata pelisivustoa.

Uutisosassa käyttäjä voi selata pelin virallisia uutisotsikoita. Mainostusosiossa näytetään pelaajien perustama ryhmä, jolla on mainostuslupa: näytettävä ryhmä arvotaan joka sivulatauksella. Yläosassa määriteltiin myös HTML-taulu, joka suljetaan sivupohjan alaosassa. Täten HTML-taulu kattaa koko pelisivun, mikä mahdollistaa pelisivujen keskittämisen pelaajan näyttöruudulle. HTML-taulun keskittäminen määriteltiin pelin tyylitiedostossa alla olevan koodin mukaisesti.

```
/* Sivupohjan taulu */
table.window_table
{
        /* Taulun leveys */
        width:720px;

        /* Sivun keskittämiseen vaadittavat määritykset*/
        margin-left: auto;
        margin-right: auto;
}
```

Jotkut sivupohjan yläosan toiminnoista vaativat tietokannan käyttöä. Näitä toimintoja varten luotiin *Header*-malli, joka välittää tietokannasta haetut tiedot *header*-näkymään. *Header*-näkymä tulkitsee nämä tiedot, ja muodostaa niiden avulla sivupohjan yläosan HTML-koodin. *Header*-näkymään luotiin myös JavaScript-funktio, jonka avulla pelaaja voi selailla uutisotsikoita reaaliaikaisesti.

5.1.2 Sivupohjan keskiosa

Sivupohjan keskiosaan määriteltiin sivukohtaiset sisällöt, jotka vaihtelevat eri pelisivuilla. Jokainen pelisivu kuitenkin noudattaa samaa teemaa, joka hahmoteltiin kuvassa 4. Pelisivujen sisältöä käsitellään tarkemmin kuudennessa luvussa.

5.1.3 Sivupohjan alaosa

Sivupohjaan lisättiin lopuksi alaosa, vaikkakaan siihen ei lisätty mitään pelaajalla näkyvää tietoa. Sivupohjan alaosaa varten luotiin *footer*-näkymä, joka voitiin tarvittaessa ladata CodeIgniterin ohjaimessa. Alaosan määriteltiin, koska se sulkee sivupohjan yläosassa luodun HTML-taulun. Täten HTML-taulun sulkemista ei tarvinnut määrittää jokaiseen pelisivuun erikseen.

5.2 Pelaajan tilatietojen reaaliaikainen päivitys

Pelissä pelaajan tilatiedot päivitetään reaaliaikaisesti Ajax-pyynnöllä, jos pelaaja suorittaa tilatietoja muokkaavan toimenpiteen, kuten esimerkiksi viitan vaihdon. Pelaajan tilatietoja esitetään alla olevassa kuvassa.

Kuva 6. Pelaajan tilatiedot pelissä.

Pelaajan tilatietoja varten tehtiin HTML-taulu nimeltään *player_status*, jonka tyyliasetuksiin määriteltiin kuvassa 6 näkyvä taustakuva. *Player_status*-taulun sisältö tyhjennetään pelaajan tilatietojen päivityksen yhteydessä. Tyhjennyksen jälkeen *player_status*-taulun sisältö rakennetaan uudelleen päivitetyillä tilatiedoilla. *Player_status*-taulun asetukset eivät kuitenkaan tyhjenny, joten *player_status*-taulun taustakuvaa ei ladata uudelleen Ajax-pyyntöjen yhteydessä.

Pelaajan tilatiedot haetaan tietokannasta, minkä vuoksi tietokantaan tehtiin taulu nimeltään *Player*. *Player*-taulun jokainen rivi vastaa yhtä pelaajaa. Nämä rivit sisältävät tietoa pelaajista, kuten esimerkiksi pelaajan rahamäärän ja tunnistenumeron. Rivit myös sisältävät useita tunnistenumeroita, jotka viittavat toisiin tauluihin tietokannassa. Esimerkiksi pelaajan ylläpitämä viitta ilmoitetaan *Player*-taulussa viitan tunnistenumerolla. Viitan tunnistenumerolla voidaan sitten hakea tarkempaa tietoa viitasta, kuten esimerkiksi viitan nimi.

Pelaajan tilatietojen päivitystä varten CodeIngiterin ohjaimeen tehtiin *player_status*-metodi, joka sallii ainoastaan Ajax-kutsuja. Alla olevassa koodissa näytetään *player_status*-metodin rakenne.

```
public function player_status()
{
        // tarkistetaan onko Ajax-pyyntö
        if ($this->input->is_ajax_request())
        {
                // Ladataan ja ajetaan "Player_status"-malli
                $this->load->model('Player_status', '', TRUE);
                $this->Player_status->main($data);

                // Ladataan "player_status"-näkymä
                $this->load->view('ajax/player_status',$data);
        }
}
```

Pelaajan tilatietojen hakemiseen tietokannasta luotiin *Player_status*-malli, joka ladataan ja ajetaan *player_status*-metodissa. *Player_status*-mallissa *$data*-muuttujan indekseihin määritetään tilatiedoissa päivitettyjä tietoja. Esimerkiksi pelaajan sijainti määritetään *$data['location']*-muuttujaan. *Player_status*-malli välittää pelaajan tilatiedot lopulta *player_status*-näkymään *$data*-muuttujan avulla.

Player_status-näkymä sisältää *player_status*-taulun sisällön. *Player_status*-mallissa välitettyjä tilatietoja käytetään *player_status*-näkymässä viittaamalla suoraan *$data*-muuttujan indekseihin. Esimerkiksi jos mallissa pelaajan sijainti on määritelty *$data['location']*-muuttujaan, näkymässä tätä tietoa voidaan kutsua *$location*-muuttujalla.

5.3 Taistelukyky

Kanien ja pelaajien taistelukyky määräytyy kolmesta attribuutista: Voimasta (*strength*), onnesta (*luck*) ja elinvoimasta (*health*). Elinvoima määrää suoraan kuinka paljon vahinkoa pelaaja tai kani kestää. Voima taas määrää kuinka paljon pelihahmo tekee vahinkoa. Onni määrittelee pelihahmon mahdollisuuden aiheuttaa vastustajalleen kriittistä, eli kaksinkertaista vahinkoa. Mitä korkeampi onni, sitä suurempi mahdollisuus kriittisiin iskuihin taistelussa. Lisäksi onni määrittelee myös mahdollisuuden ohi-iskuihin, jotka eivät tuota ollenkaan vahinkoa vastustajalle taistelussa. Ohi-iskujen todennäköisyys laskee, kun pelihahmon onni suurenee. Tämän lisäksi onni vaikuttaa siihen, kumpi osapuoli saa aloittaa taistelun. Pelaaja kykenee korottamaan kolmea attribuuttiaan käyttämällä loitsuja, sauvoja, viittoja ja keräämällä lisäominaisuuksia (*perks*).

Pelaajan elinvoimapisteet lasketaan summaamalla kaikki elinvoimapisteitä antavat lisäominaisuudet ja varusteet, joita pelaaja käyttää sillä hetkellä. Pelaajan voimapisteet ja onni lasketaan samalla periaatteella kuin elinvoimapisteet. Pelaajien ylläpitämät varusteet määritellään tietokannan *Player*-taulun riveissä. Nämä rivit sisältävät varusteiden tunnistenumerot, joiden avulla haetaan pelaajan ylläpitämien varusteiden tiedot tietokannasta. Jokaiselle varustetyypille tehdään tietokantaan oma taulu, jossa yksi rivi kuvaa yhtä varustetta.

Pelaajat voivat ostaa lisäominaisuuksia, tai pelaajat voivat ansaita niitä voittamalla tiettyjä kaneja. Pelaajaryhmiin kuuluvat voivat ostaa pelaajaryhmille tarkoitettuja lisäominaisuuksia, jotka nostavat ryhmän jäsenten taistelukykyjä. Pelaajien lisäominaisuuksia varten luotiin oma taulu nimeltään *Perk*, joka sisältää pelaajien ansaitsemat lisäominaisuudet. Tässä taulussa yksi rivi kuvaa yhden pelaajan yhtä lisäominaisuutta. Rivit sisältävät tietoja lisäominaisuuksista, kuten lisäominaisuuden antamat elinvoima- ja voimapisteet.

Kaneja varten luodaan oma taulu, jonka yksi rivi kuvaa yhtä kania. Riveissä määritellään muun muassa kanien sijainti ja yleisyys. Mitä korkeampi yleisyys kanilla on, sitä useammin pelaajat kohtaavat sen taistelussa.

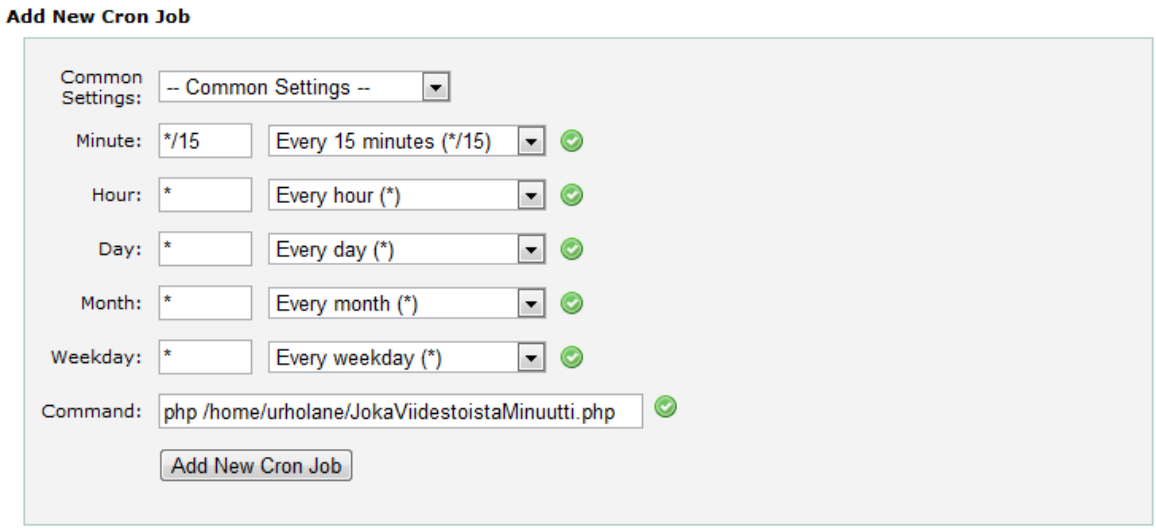
5.4 Taistelusysteemi

Pelin taistelut tapahtuvat tekstipohjaisesti, ja ne generoidaan automaattisesti osapuolien taistelukykyjen perusteella. Taistelut käydään osapuolien iskiessä toisiaan vuorotellen, kunnes toisen elinvoima laskee nollaan. Käytyjen taisteluiden tulokset näkyvät pelaajien lokeissa. Pelaajalla on mahdollisuus taistella kolmea eri tyyppistä taistelua:

- normaali taistelu yhtä kania vastaan
- kaksintaistelu toista pelaajaa vastaan
- ryhmätaistelu pelaajaryhmässä yhtä tai useampaa kania vastaan.

Taistellessaan kaneja vastaan pelaajalta kuluu 10 kestävyyspistettä (*stamina*). Pelaajien kestävyyspisteet nousevat automaattisesti joka viidestoista minuutti kymmenellä kestävyyspisteellä; kestävyyspisteiden maksimimäärä on 255. Pelaajan kestävyyspisteet näkyvät pelaajan tilatiedoissa.

Pelaajien kestävyyspisteiden automaattinen lisäys joka viidestoista minuutti tehtiin Cronin avulla. Peliprojektissa käytettävä web-hotelli tarjosi selkeän käyttöliittymän (ks. kuva 7) Cron-töiden suorittamiseen.



Kuva 7. Cron-töiden lisäämiseen tarkoitettu käyttöliittymä.

Kuvassa 7 on määritelty *JokaViidestoistaMinuutti.php*-tiedoston ajo joka viidestoista minuutti. Tämä tiedosto sijaitsee CodeIgniterin ulkopuolella, joten siihen määritettiin tietokannan käyttöön vaadittavat asetukset. Asetuksien jälkeen tiedostoon määriteltiin SQL-lause, joka lisää kaikkien pelaajien kestävyyspisteitä kymmenellä pisteellä.

Pelaaja voi suorittaa normaalin taistelun, eli taistella yksittäistä kania vastaan, jos *Fight*-nappi on aktiivinen, eli keltaisen värinen. *Fight*-napin aktiivisuus riippuu pelaajan sijainnista ja kestävyyspisteistä. Pelimaailman useimmat matkustettavat kohteet mahdollistavat pelaajan normaalin taistelun. Joissakin kohteissa ei ole kaneja, mikä estää normaalien taistelujen suorittamisen. *Fight*-nappia painaessa pelaajalle arvotaan kani vastustajaksi. Joidenkin kanien esiintyminen taisteluissa on todennäköisempää kuin toisten kanien. Normaalin taistelun alkaessa arvotaan aloittaja, minkä jälkeen osapuolet iskevät vuorotellen. Iskuja suoritetaan kunnes toisen osapuolen elinvoimapisteet loppuvat. Voittamalla normaalin taistelun pelaaja saa kultaa ja kokemusta: Lisäksi pelaajalla on mahdollisuus saada taikoja, sauvoja, viittoja tai lisäominaisuuksia. Taistelun tulos ilmoitetaan pelaajan lokissa, josta pelaaja voi tarkastella taistelun yksityiskohtia.

Pelissä pelaaja pystyy lähettämään kaksintaisteluhaasteen toiselle pelaajalle. Haastaja voi myös valita liitetäänkö haasteeseen kultavaatimus, jonka molempien osapuolien tulee maksaa. Kaksintaistelun voittaja lunastaa molempien pelaajien suorittamat maksut. Jos haastettavalla on riittävästi kultaa, ja jos hän hyväksyy haasteen, kaksintaistelu voidaan aloittaa. Kaksintaistelussa arvotaan ensiksi aloittaja, minkä jälkeen osapuolet suorittavat iskuja vuorotellen, kunnes toisen elinvoimapisteet loppuvat. Kaksintaistelun tulos ilmoitetaan molempien pelaajien lokeissa, joissa pelaajat voivat tarkastella taistelun yksityiskohtia.

Pelimaailma sisältää pelaajaryhmille tarkoitettuja kohteita, titaanilaaksoja, joissa pelaajaryhmät voivat yhdessä taistella pelin voimakkaimpia titaanikaneja vastaan. Kaikki pelaajat voivat matkustaa näihin kohteisiin, ja taistella siellä olevia kaneja vastaan, mutta tätä ei suositella titaanikanien voimakkuuden vuoksi. Pelaajan matkustaessa titaanilaaksoon *Fight*-nappi muuttuu *Party fight*-napiksi. Jos *Party fight*-nappi on aktiivinen, eli keltaisen värinen, pelaaja voi aloittaa taistelun. Jos pelaajalla ei ole vähintään 10 kestävyyspistettä, *Party fight*-nappi deaktivoituu, ei-

kä sitä voi klikata. Pelaajan klikatessa *Party fight* -nappia, peli tunnistaa automaattisesti onko pelaajan mahdollisen pelaajaryhmän muita jäseniä samassa kohteessa, ja liittää ne mukaan taisteluun, jos heillä on vähintään 10 kestävyyspistettä. Taistelu etenee jokaisen pelaajan iskiessä vuorotellen ensiksi, minkä jälkeen titaanikani iskee satunnaisesti yhtä pelaaja, minkä jälkeen elossa olevat pelaajat iskevät taas. Tätä jatketaan, kunnes toisen osapuolten jäsenten elinvoimapisteet laskevat nollaan. Titaanikanien voittamisesta saa suuremman määrän kultaa, kokemusta ja paremman mahdollisuuden saada varusteita verrattuna muihin kaneihin. Taistelun tulokset ja yksityiskohdat julkistetaan taisteluun osallistuneiden pelaajien lokeissa.

Taistelut suoritetaan Ajax-pyynnöllä, joka linkitetään sivupohjan alaosaan, joten pelaaja voi suorittaa taistelun reaaliaikaisesti jokaisessa pelisivussa. Jos pelaaja suorittaa taistelun ollessaan pelin etusivulla, pelaaja ohjataan reaaliaikaisesti etusivun lokiin. Sen jälkeen taistelun tulos ilmaantuu lokiin JavaScript-animaationa, ja pelaajan tilatiedot päivitetään. Jos pelaaja suorittaa taistelun ollessaan muualla kuin pelin etusivulla, pelaajaa ei ohjata toiselle sivulle.

Taisteluja varten luotiin *Fight*-metodi CodeIgniterin ohjaimeen. *Fight*-metodi laitettiin hyväksymään ainoastaan Ajax-pyyntöjä. *Fight*-metodissa ladataan ja ajetaan *Fight*-malli. *Fight*-malli tarkistaa onko pelaajan mahdollista suorittaa taistelu. Jos taistelun suorittaminen on mahdollista, *Fight*-malli generoi taistelun. Taistelun generoimisen aikana jokainen iskun jälkeinen tilanne tallennetaan tietokantaan. Tietokantaan tallennetaan muun muassa kumpi osapuoli suoritti iskun ja kuinka paljon vastustajalle jäi elinvoimapisteitä iskun jälkeen. Iskuja varten luotiin *Strikes*-taulu tietokantaan.

5.5 Pelin lokalisointi

Pelaajalle tehtiin mahdolliseksi valita pelikieleksi englanti tai suomi, minkä toteuttamiseen sovelletaan CodeIgniterin kielikirjastoa. Kielitiedostot sijaitsevat omissa kansioissaan *application/language*-kansiossa, jonne lisätään *english*-kansion lisäksi *finnish*-kansio. Näihin kansioihin luodaan tiedostoja, jotka sisältävät kaikki käyttäjälle näkyvät tekstit molemmilla kielillä. Molempiin kielikansioihin määritetään

kielitiedostot web-sivujen mukaan, esimerkiksi pelin etusivulla näkyvät suomen- ja englanninkieliset tekstit (pois lukien virheviestit) tallennetaan kahteen *index*-nimisiin kielitiedostoihin, jotka sijoitetaan *english*- ja *finnish*-kansioihin. Tämän lisäksi luodaan kielitiedostot virheviestejä varten. Kielitiedostot koostuvat indeksoiduista muuttujista, jotka nimetään *$lang*-muuttujiksi. *$lang*-muuttujat sisältävät tekstiä, ja muuttujien indekseissä määritetään miten kyseistä muuttujaa voidaan hakea. Seuraavassa koodissa esitetään osa suomenkielisen *index*-kielitiedoston rakenteesta.

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

$lang['news'] = "uutiset";
$lang['battle_victory_header'] = "Onneksi olkoon, voitit taistelun!";
$lang['Good_day_user'] = "Hyvää päivää %s";
```

Yllä olevassa koodissa ensimmäisellä rivillä estetään suora pääsy tiedostoon verkkoselaimen kautta. Seuraavissa riveissä määritellään *$lang*-muuttujia ja niiden indeksejä. Viimeisessä rivissä *%s* tarkoittaa, että kyseistä tekstiä kutsuttaessa tulee välittää myös merkkijono-tyyppinen parametri.

$lang-muuttujia käytetään ensiksi lataamalla haluttu kielitiedosto, minkä jälkeen kutsutaan haluttu muuttuja. Kielitiedostoa ladattaessa määritellään mitä kieltä käytetään. Kielen määritys tapahtuu ohjaimen konstruktorissa, joka päivittää käyttäjän valitseman kielen asetuksiin. Alla olevassa koodissa näytetään esimerkki kielitiedoston latauksesta ja käytöstä.

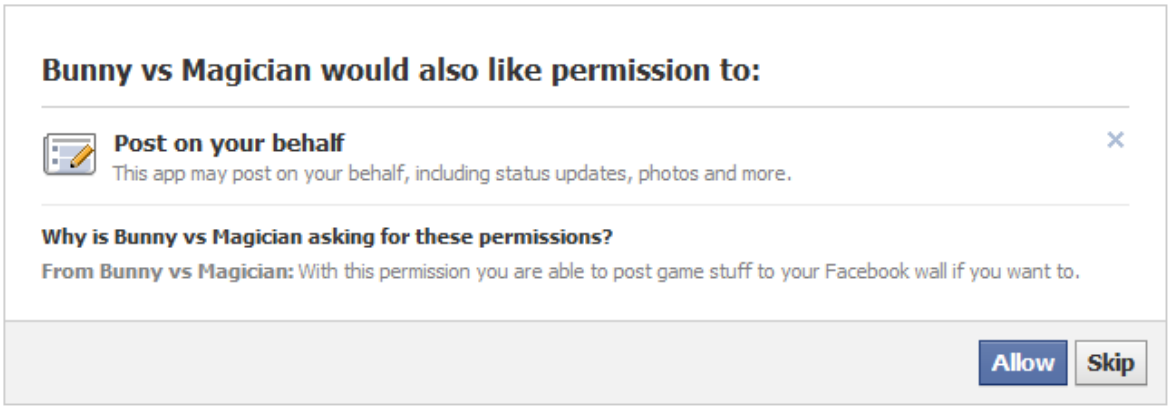
```
$this->lang->load('index', 'finnish');  //ladataan kielitiedosto
$text1 = $this->lang->line('news');
$text2 = $this->lang->line('Good_day_user', 'Pelaaja');
```

Yllä olevassa koodissa ensimmäisessä rivissä ladataan suomenkielinen kielitiedosto nimeltään *index*. Toisessa rivissä haetaan teksti, jonka indeksinä on *news*. Kolmannessa rivissä haetaan teksti, jonka indeksi on *Good_day_user*. Tämän tekstin haun yhteydessä välitetään myös parametri, jonka arvo on *Pelaaja*. Haetut tekstit varastoidaan *text1*- ja *text2*-muuttujiin.

6 PELISIVUT

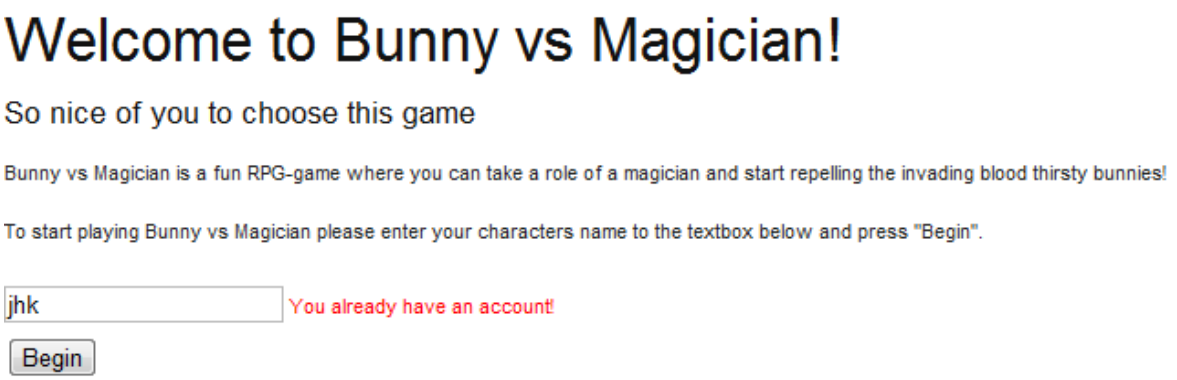
6.1 Rekisteröintisivu

Peliin rekisteröityminen tapahtuu Facebookin kautta, ja pelin rekisteröintiin liittyvät asetukset määritellään Facebook Developerin web-sivuilla. Asetuksissa määritellään, mitä oikeuksia sovellus pyytää käyttäjältä rekisteröinnin yhteydessä käyttäjän ei kuitenkaan tarvitse suostua kaikkiin pyyntöihin. Peliprojektissa pelin rekisteröinnin yhteyteen määriteltiin pyydettävä lupa (ks. kuva 8), jonka pelaaja voi halutessaan torjua. Tämän luvan avulla peli kykenee lähettämään viestejä käyttäjän Facebook-seinälle. Peliä pelattaessa näitä viestejä ei koskaan lähetetä ilman käyttäjän suostumusta.



Kuva 8. Pelin rekisteröinnin yhteydessä kysyttävä lisälupa.

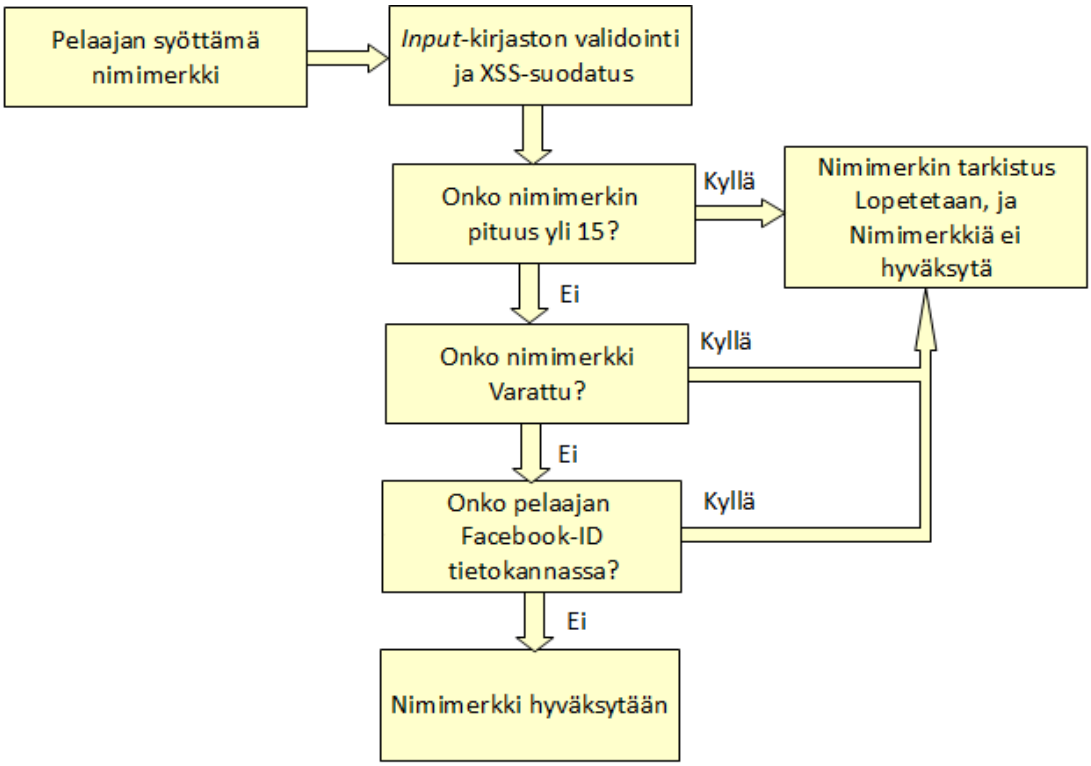
Kun pelaaja on vastannut esitettäviin pyyntöihin, hänet ohjataan rekisteröintisivulle (ks. kuva 9), jossa pelaaja saa päättää roolihahmonsa nimimerkin. Kun pelaaja on kirjoittanut haluamansa nimimerkin, koodi tarkistaa onko kirjoitettu nimimerkki varattu, ja ilmoittaa tarkistuksen tuloksen käyttäjälle reaaliaikaisesti. Jos pelaajan kirjoittama nimimerkki ei ole varattu, nappi aktivoituu, ja nappia painamalla pelaaja ohjataan pelin etusivulle.



Kuva 9. Pelin rekisteröintisivu.

Pelin jokaisen sivulatauksen yhteydessä tarkistetaan löytyykö pelaajan Facebook-ID tietokannan taulusta. Jos pelaajaa ei löydy, hänet ohjataan rekisteröintisivulle. Edellä mainittu tarkistus koostuu yhdestä malli-tiedostosta, joka suoritetaan CodeIgniterin ohjaimen konstruktorissa. Koska mallia käytetään joka sivulatauksella, se määritetään automaattisesti latautuvaksi, jolloin sitä ei tarvitse ladata erikseen ohjaimen konstruktorissa. Tämä määriteltiin CodeIgniterin *autoload.php*-tiedostossa, joka sijaitsee *application/config*-kansiossa.

Pelin rekisteröintisivun tärkein elementti on tekstikenttä, johon pelaaja voi kirjoittaa pelihahmonsa nimimerkin. Pelaajan muokatessa tekstikentän sisältöä jQuery-funktio aktivoituu ja lähettää Ajax-kutsun palvelimelle. Ajax-kutsu vastaanotetaan CodeIgniterin ohjaimessa, jossa määritetään malli, joka validoi syötetyn nimimerkin. Mallissa nimimerkki validoidaan ensiksi käyttäen CodeIgniterin *input*-kirjastoa ja XSS-suodatusta. Tämän jälkeen tarkistetaan nimimerkin pituus (kirjainten maksimimäärä on 15), minkä jälkeen tarkistetaan onko nimimerkki jo varattu. Viimeisenä tarkistetaan onko pelaajalla jo pelitili. Kuvassa 10 näytetään syötetyn nimimerkin validointiprosessi. Jos pelaajan syöttämä nimimerkki hyväksytään, koodi aktivoi napin, jota painamalla nimimerkki tarkistetaan vielä kerran. Jos nimimerkki hyväksytään toisenkin tarkistuksen jälkeen, pelaajalle luodaan pelitili.



Kuva 10. Pelaajan nimimerkin validointi.

Pelitilin luomista varten tietokantaan tehdään *Pelaaja*-taulu, johon lisätään uusi rivi jokaista pelitiliä varten. Jokaisessa rivissä määritellään yksittäisen pelitilin tietoja, kuten esim. Facebook-ID, tilin luomispäivämäärä, sijainti ja nimimerkki. Pelaaja-taulusta indeksoidaan Facebook-ID-sarake.

6.2 Etusivu

Pelin etusivusta kehitettiin pelaajien profiilisivuksi, joka tarjoaa pelaajalle tietoa roolihahmostaan. Etusivulla pelaajat pystyvät suorittamaan useita toiminnallisuuksia, kuten esimerkiksi varusteiden vaihtoja. Pelaajat pystyvät myös katselemaan toisten pelaajien profiilisivuja. Pelin etusivu muodostui kuvan 11 näköiseksi.

Kuva 11. Pelin etusivun ulkoasu.

Etusivu sisältää useita alalinkkejä, joita kuvataan keltaisilla välilehdillä. Välilehtien sisällöt ladataan etusivun latauksen yhteydessä, minkä jälkeen niiden näkyvyyttä kontrolloidaan JavaScriptillä. Etusivua varten luotiin:

- *index*-niminen metodi CodeIgniterin ohjaimeen
- *Index*-niminen malli
- index-niminen näkymä.

Index-metodin tehtävä on vastata etusivun sivupyyntöön lataamalla tarvittavat tiedostot pelin etusivun rakentamiseen. Seuraavassa koodissa näytetään *index*-metodin rakenne.

```
function index()
{
        //Ladataan ja ajetaan sivupohjan yläosan malli
        $this->load->model('Header', '', TRUE);
        $this->Header->main($data);

        //Ladataan ja ajetaan etusivun malli
        $this->load->model('Index', '', TRUE);
        $this->Index->main($data);

        //Ladataan sivupohjan yläosan näkymä
        $this->load->view('templates/header', $data);

        //Ladataan etusivun näkymä
        $this->load->view('pages/index', $data);

        //Ladataan sivupohjan alaosan näkymä
        $this->load->view('templates/footer');
}
```

Yllä olevassa koodissa ensiksi ladataan ja ajetaan *Header*-niminen malli, joka sisältää sivupohjan yläosan tietoja. Tämän jälkeen ladataan ja ajetaan *Index*-malli, joka sisältää etusivun tietoja. Molemmat mallit varastoivat etusivun luomiseen tarvittavat tiedot *$data*-muuttujaan. *$data*-muuttuja välitetään lopulta näkymiin, jotka ladataan yllä olevan mukaisesti.

Index-malli tehtiin hoitamaan etusivun personalisoinnin, eli pelaajan tietojen hakemisen tietokannasta.

Index-näkymään tehtiin etusivun HTML-koodin lisäksi JavaScript-funktiot, joilla kontrolloidaan alalinkkien sisältöjen näkyvyyttä.

6.2.1 Profiilikuva

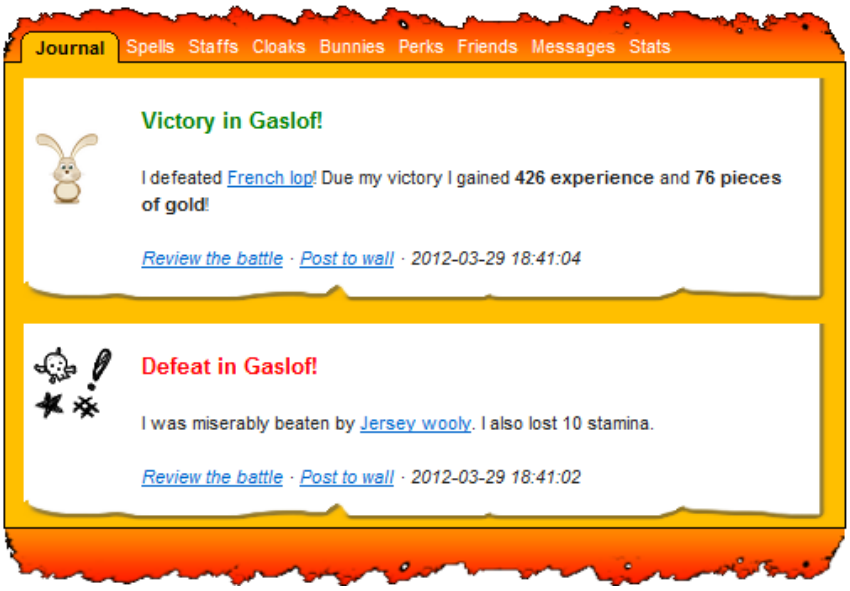
Profiilisivulle tehtiin profiilikuva, josta ilmenee kenen pelaajan profiilisivua selataan. Kuvassa 12 esitellään tehtyä profiilikuvaa.

Kuva 12. Pelin profiilikuva.

Profiilikuvassa näkyy myös pelaajan nimimerkki ja linkki profiilikuvan vaihtoon. Profiilikuvan vaihto suoritetaan reaaliaikaisesti Ajaxilla käyttäen CodeIgniterin tiedostojen lataus kirjastoa. Profiilikuvan vaihdossa pelaajan valitsema kuva ladataan palvelimelle, ja tietokannassa päivitetään pelaajan uuden profiilikuvan nimi. CodeIgniterin tiedostojen latauskirjasto nimeää kuvan uudelleen, jos palvelimelta löytyy saman niminen kuvatiedosto.

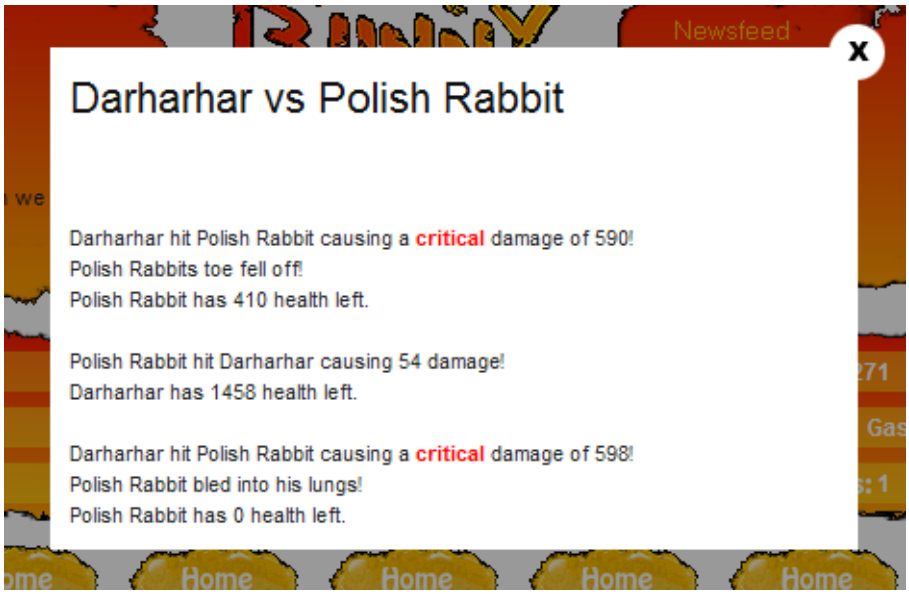
6.2.2 Loki

Pelin etusivulle tehtiin loki, jossa näkyy pelaajan kymmenen uusinta taistelua. Kuvassa 13 näkyy kaksi lokimerkintää.



Kuva 13. Pelaajan loki.

Ylemmässä lokimerkinnässä pelaaja on voittanut taistelun kania vastaan. Alemmassa lokimerkinnässä pelaaja on hävinnyt taistelun kania vastaan. Pelaaja voi tarkastella taisteluiden yksityiskohtia klikkaamalla kuvassa 13 näkyviä *Review the battle* -linkkejä. Alla olevassa kuvassa näytetään taistelun yksityiskohtia.

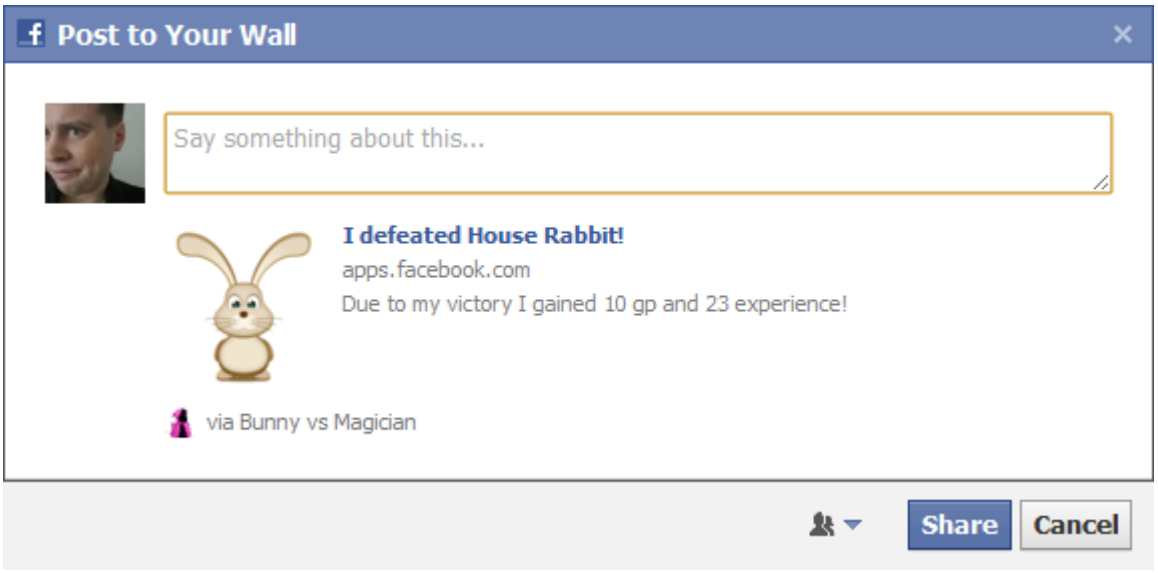


Kuva 14. Taistelun yksityiskohtien tarkastelua.

Yllä olevassa kuvassa näkyy kuinka osapuolet iskevät toisiaan vuorotellen. Pelaaja nimimerkiltään *Darharhar* iskee ensiksi kriittisen iskun, minkä jälkeen kani iskee

pelaajaa. Lopuksi pelaaja iskee toisen kriittisen iskun, jolla pelaaja voittaa taistelun.

Pelaajat voivat myös julkaista taistelun tuloksia omille Facebook-seinilleen klikkaamalla kuvassa 13 näkyviä *Post to wall* -linkkejä. Klikkaamalla tälläistä linkkiä avautuu alla olevan kuvan mukainen esikatseluikkuna, jonka *Share*-nappia klikkaamalla pelaaja voi julkaista taistelun Facebook-seinälleen.



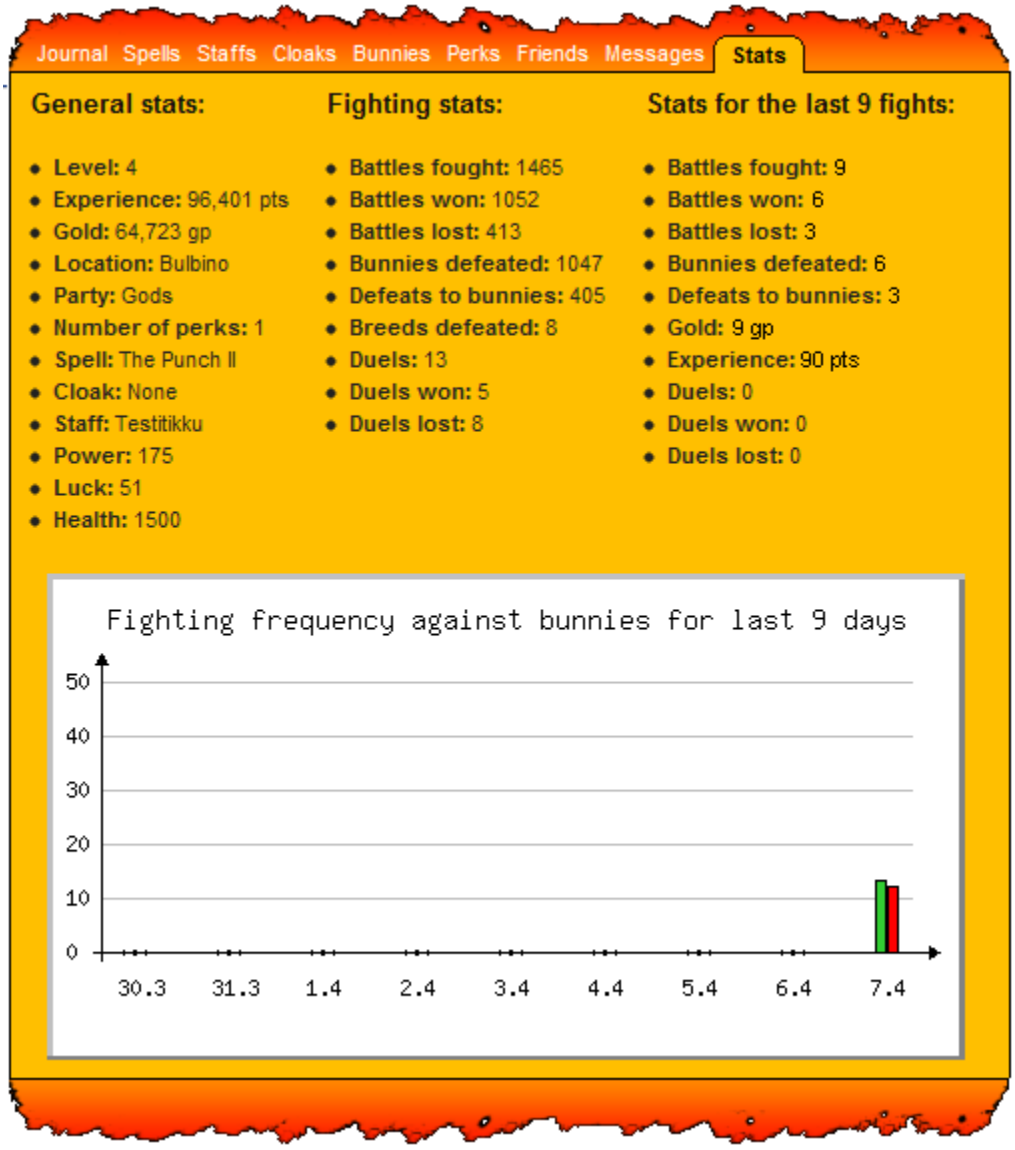
Kuva 15. Viestin lähettäminen pelaajan Facebook-seinälle.

Pelaajien lokimerkintöjä varten tietokantaan luotiin taulu, jonka jokainen rivi vastaa yhtä lokimerkintää. Nämä rivit sisältävät kaiken tarpeellisen tiedon lokimerkinnän julkaisemiseen. Lokimerkintöjen tiedot välitetään *index*-näkymään, jossa tiedot prosessoidaan. Lopulta tuloksena saatiin kuvan 13 mukaisia lokimerkintöjä. Jos pelaajalle on kertynyt yli kymmenen lokimerkintää, *Index*-malli poistaa pelaajan vanhimpia lokimerkintöjä tietokannasta.

Taistelun tuloksien lähettäminen pelaajan Facebook-seinälle tehtiin Facebookin tarjoamalla JavaScript-kirjastolla. Kirjasto ladattiin *index*-näkymään, minkä jälkeen luotiin *Feed Dialog* -funktio. *Feed Dialog* -funktion tarkoitus on muodostaa kuvan 15 näköisiä esikatseluikkunoita. *Feed Dialog* -funktion koodi saatiin Facebook Developerin sivuilta. Alkuperäistä *Feed Dialog* -funktiota muokattiin, jotta pelaaja pystyisi lähettämään viestejä eri tyylisten taistelujen lopputuloksista. *Feed Dialog* -funktioon asetettiin myös parametrejä, jotka määrittelevät taistelun yksityiskohtia.

6.2.3 Pelaajatilasto

Pelin etusivulla pelaaja voi katsella oman tai muiden roolihahmojen tilastoja. Tilastoista näkyy muun muassa pelaajan taistelusuoritukset ja ylläpitämät varusteet. Alla olevassa kuvassa esitellään etusivun pelaajatilastoa.



Kuva 16. Pelaajatilastot.

Yllä olevan kuvan pylväsdiagrammissa kuvataan pelaajan yhdeksää viimeistä taistelua kaneja vastaan. Vaaka-akselissa ilmoitetaan päiväykset ja pystyakselissa taistelumäärät. Vihreä pylväs kuvaa voitettuja taisteluja ja punainen hävittyjä taisteluja. Pylväsdiagrammi koostuu PHP-tiedostosta, jolle välitetään pelaajan Face-

book-ID. Tämän jälkeen tiedosto lataa pelaajan tarvittavat tiedot tietokannasta ja piirtää edellä olevan kuvan näköisen pylväsdiagrammin.

6.2.4 Inventaario

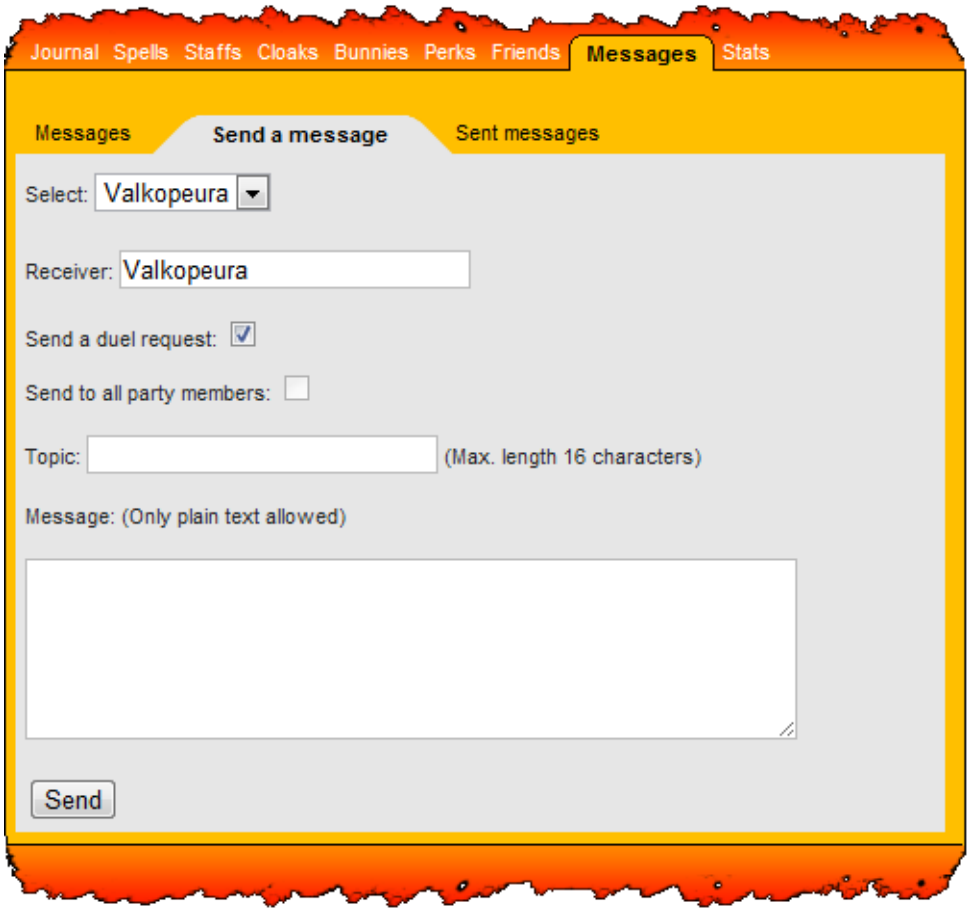
Pelin etusivulla pelaaja pystyy tarkastelemaan oman roolihahmonsa tai muiden roolihahmojen inventaarioita. Inventaario sisältää pelaajan taiat, viitat, taikasauvat ja lisäominaisuudet. Inventaariossa pelaaja pystyy myös vaihtamaan roolihahmonsa ylläpitämiä varusteita. Alla olevassa kuvassa esitellään pelaajan inventaariota, joka on suodatettu näyttämään pelaajan taiat.

Kuva 17. Pelaajan taiat.

Yllä olevassa kuvassa näkyy kaksi taikaa ja niiden tietoja. Taikoja kuvataan vasemmalla näkyvissä kuvissa (joiden viimeistely on vielä kesken). Pelaaja voi vaihtaa ylläpitämäänsä taikaa klikkaamalla oikealla näkyviä nappeja.

6.2.5 Viestit

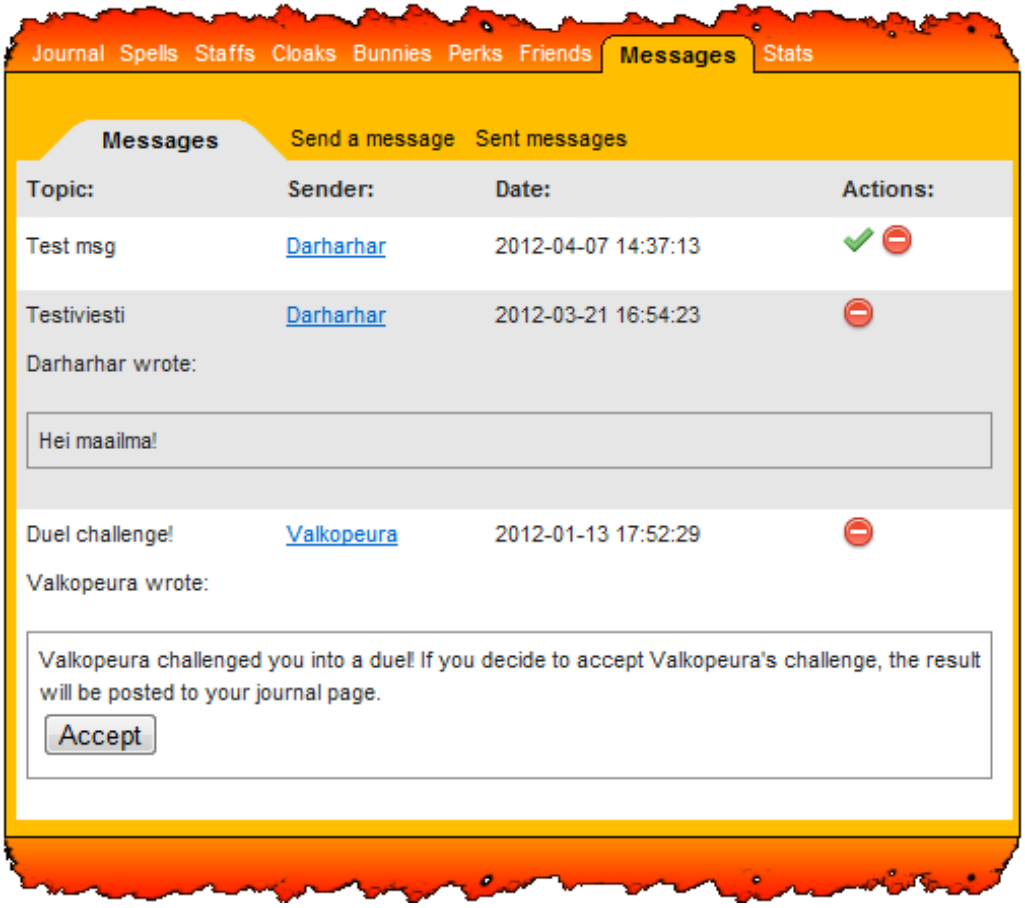
Pelin etusivulla pelaajilla on mahdollisuus lähettää toisille pelaajille viestejä. Viestit voivat myös olla kaksintaisteluhaasteita. Jos pelaaja kuuluu ryhmään, pelaaja voi lähettää viestin jokaiselle ryhmän jäsenelle. Viestit näkyvät ainoastaan pelaajan omalla profiilisivulla. Alla olevassa kuvassa esitellään viestin lähettämiseen tarkoitettu käyttöliittymä.



Kuva 18. Viestien lähetyksiin tarkoitettu käyttöliittymä.

Viestin lähetyksen jälkeen pelaaja ohjataan *Sent messages* -välilehdelle, jossa pelaaja voi tarkistaa menikö viesti perille. Viestien lähetykset suoritetaan reaaliaikaisesti Ajaxilla.

Vastaanotettuja viestejä varten luotiin kuvan 19 mukainen käyttöliittymä.



Kuva 19. Pelaajan saapuneet viestit.

Yllä olevassa kuvassa näkyy kolme viestiä: Ylin viesti on avaamaton, seuraava on avattu viesti ja viimeinen on avattu kaksintaisteluhaaste. Avaamattomien viestien lukumäärä ilmoitetaan pelaajan tilatiedoissa. Aukaistu viesti merkitään automaattisesti luetuksi. Pelaaja voi myös suoraa klikata *Check*-ikonia merkitäkseen viestin luetuksi. Viestejä näkyy maksimissaan kaksikymmentä kerralla. Viestejä voi poistaa klikkaamalla punaista ikonia. Kaikki edellä mainitut toiminnot suoritetaan reaaliaikaisesti.

6.2.6 Kaverit

Pelaaja voi pelin etusivulla tarkastella niitä Facebook-kavereitaan, jotka ovat rekisteröityneet peliin. Pelaajat voivat tarkastella ainoastaan omia kavereitaan. Pelaaja voi nopeasti huomata missä päin pelimaailmaa kaverit ovat. Kuvassa 20 esitellään tähän tarkoitukseen toteutettua käyttöliittymää.

Kuva 20. Pelaajan kaverit.

Yllä olevassa kuvassa näkyy että pelaajan kolme kaveria ovat pelimaailman kohteessa nimeltään *Gaslof*. Jos pelaaja kohdistaa kursorin kaverinsa kuvaan, hänen nimimerkkinsä tulee näkyviin. Klikattaessa kaverin kuvaa pelaaja ohjataan hänen profiilisivulle.

6.2.7 Kanit

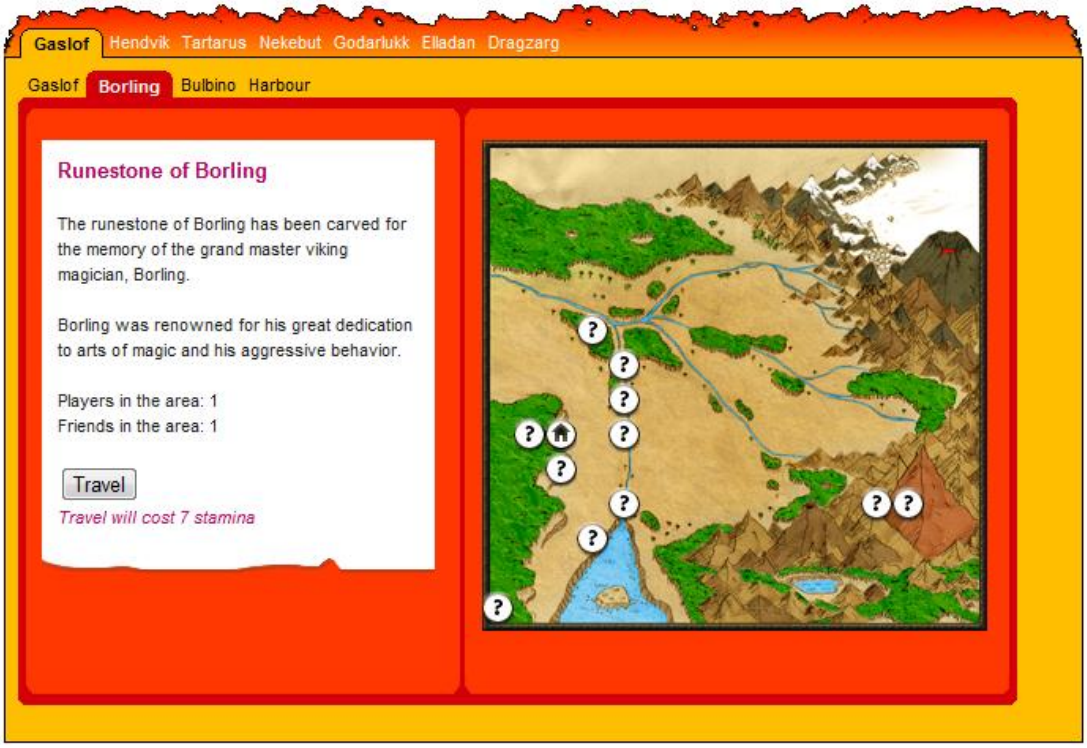
Pelin etusivulla pelaaja voi tarkastella niitä kaneja, jotka pelaaja on voittanut taistelussa. Tätä varten tehtiin kuvan 21 näköinen käyttöliittymä.

Kuva 21. Pelaajan voitetut kanit.

Yllä olevassa kuvassa näkyy kahdeksan eri lajin kania (kanien kuvat ovat vielä tekeillä), jotka pelaaja on voittanut taistelussa. Kanien nimien alla ilmoitetaan kuinka monesti pelaaja on voittanut ne taistelussa. Kanit lajitellaan niiden taistelukykyjen mukaan. Korkeimman taistelukyvyn omaava kanilaji luetellaan ensimmäiseksi.

6.3 Matkustus-sivu

Pelaaja kykenee matkustamaan pelimaailmassa, jota kuvataan kartalla. Pelaaja kykenee matkustamaan haluamaansa kohteeseen, jos pelaajan hahmolla on riittävästi kestävyyspisteitä. Kahden kohteen väliselle matkalle vaadittava kestävyysmäärä lasketaan pelaajan nykyisen ja matkustettavan kohteen välillä. Mitä kauempana kohteet sijaitsevat toisistaan pelimaailmassa, sitä enemmän pelaajalta vaaditaan kestävyyspisteitä matkustamiseen. Matkustamiseen käytettävät toiminnot suoritetaan reaaliaikaisesti JavaScriptillä tai Ajax-tekniikalla. Matkustus-sivu esitetään kuvassa 22.



Kuva 22. Pelin matkustus-sivun ulkoasu.

Yllä olevassa kuvassa näkyvä keltainen välilehti (jossa lukee *Gaslof*) kuvastaa valittua maa-aluetta. Välilehdet vierekkäiset tekstit kuvastavat muita maa-alueita, joita pelaaja voi valita. Keltaisen välilehden alla näkyy ne kohteet, jotka sijaitsevat valitulla maa-alueella. Jos pelaaja valitsee jonkin näistä kohteista, kohde muuttuu punaiseksi välilehdeksi. Pelaaja voi myös valita kohteita klikkaamalla kartalla näkyviä ympyröitä.

Matkustaakseen pelaajan on klikattava yllä olevassa kuvassa näkyvää *Travel*-painiketta. *Travel*-painike ilmaantuu ainoastaan silloin kun matkustaminen on mahdollista.

Matkustus-sivu rakentuu samalla periaatteella kuin pelin etusivukin. CodeIgniterin ohjaimeen luotiin metodi nimeltään *travel*, jota kuvataan seuraavassa koodissa.

```
function travel()
{
        // Ladataan ja ajetaan sivupohjan yläosan malli
        $this->load->model('Header', '', TRUE);
        $this->Header->main($data);

        //Ladataan ja ajetaan matkustus-sivun malli
        $this->load->model('Travel', '', TRUE);
        $this->Travel->main($data);

        //Ladataan sivupohjan yläosa
        $this->load->view('templates/header', $data);

        //Ladataan sivupohjan keskiosa
        $this->load->view('pages/travel', $data);

        //Ladataan sivupohjan alaosa
        $this->load->view('templates/footer');
}
```

6.3.1 Pelimaailma

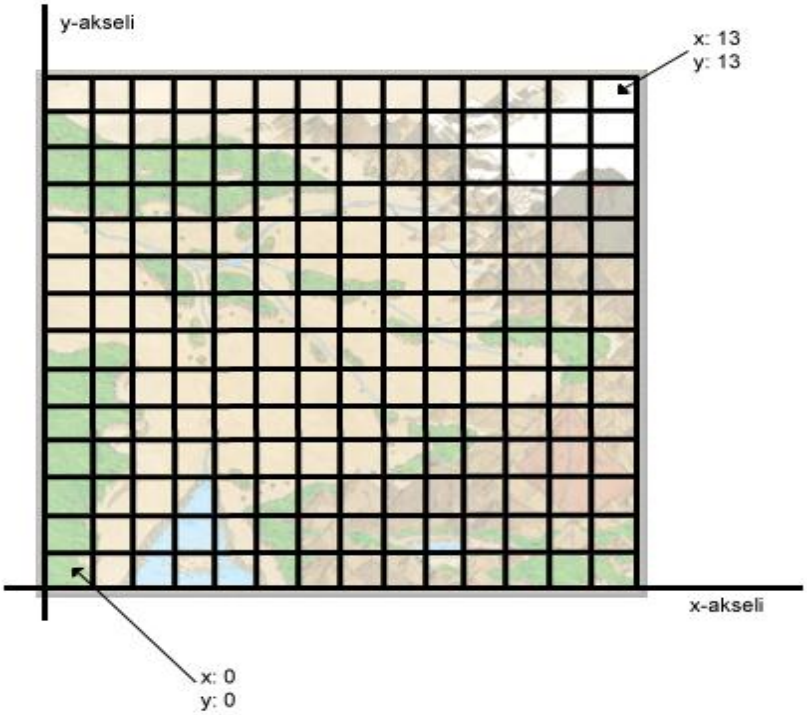
Pelimaailmaa kuvataan kartalla, joka tehtiin käyttäen tekijän omistamaa Photoshop CS5 -lisenssiä. Alla olevassa kuvassa näytetään pelimaailmaa kuvaavaa karttaa.

Kuva 23. Pelimaailman kartta.

Yllä oleva kartta toimii pohjana pelimaailman sijainneille, joita varten tietokantaan määriteltiin taulu nimeltään *Location*. *Location*-taulun jokainen rivi vastaa yhtä

kohdetta pelimaailmassa. Jokaiseen kohteeseen määriteltiin x- ja y-koordinaatit, joiden avulla kohteet sijoitetaan kartalle.

Pelimaailman kohteet sijoitetaan kartalle, kun matkustus-sivu ladataan. Tätä varten kartta määritettiin HTML-taulukon (nimeltään *map*) taustakuvaksi. *Map*-taulukkoon tehtiin koordinaatisto, jota hahmotellaan kuvassa 24.



Kuva 24. Koordinaatiston hahmottelua pelimaailman karttaan.

Yllä olevassa kuvassa hahmoteltu koordinaatisto ohjelmoitiin *travel*-näkymässä seuraavan koodin mukaisesti.

```
<table id="map">
        <?php
                for($i=13;$i>=0;$i--) //rivit
                {
                        echo "<tr>";
                        for($j=0;$j<13;$j++) //sarakkeet
                        {
                                echo "<td id='".$j.$i."'
                                x='$j' y='$i' class='map_cell'></td>";
                        }
                        echo "</tr>";
                }
        ?>
<tr><td/></tr>
```

Yllä olevassa koodissa ensimmäiseksi luodaan HTML-taulukko nimeltään *map*, jonka sisältö ohjelmoidaan PHP:llä. *map*-taulukkoon luodaan ensiksi 13 riviä, ja jokaiseen riviin luodaan 13 saraketta. Lopulta saadaan 169 solua, joilla on uniikit x- ja y-koordinaatit.

Kun matkustus-sivua ladataan, matkustus-sivua varten luotu *Travel*-malli välittää pelimaailman kohteiden tiedot *travel*-näkymään. Kohteiden tiedot prosessoidaan *travel*-näkymän JavaScript-funktiossa. Kohteiden koordinaattien perusteella *map*-taulukosta valitaan koordinaatteja vastaavat solut, minkä jälkeen pelimaailman kohteet sijoitetaan niihin.

Kartan koordinaatistojärjestelmän etuna on joustavuus. Jos karttaa halutaan jostain syystä siirtää, kartan kohteet pysyvät ”oikeissa” kohdissa. Koordinaatistojärjestelmän etuna on myös se, että sillä voidaan helposti sijoittaa pelimaailman kohteet haluttuun kohtaan kartassa.

7 TULOKSET

Suurin osa tavoitteista saavutettiin. Päätavoite osoittautui työn alkuvaiheessa jo hyvin työlääksi, joten oli odotettua ettei sitä saavutettaisi täysin. Koska pelistä haluttiin visuaalisesti miellyttävä, pelin toteuttamisen vaikeimmaksi osuudeksi osoittautui pelin graafisten elementtien teko ja käyttöliittymien suunnittelu. Vaikeuksia tuotti myös joidenkin peliominaisuuksien monimutkaiset rakenteet ja niiden toimivuus. Pelin kehitys saatiin kuitenkin pitkälle, eikä pelin valmistuminen jäänyt kaukaiseksi haaveeksi. Peliin saatiin tehtyä:

- ulkoasullinen teema
- pelijuoni
- Facebook-integraatio
- useita pelaamiseen tarvittavia ominaisuuksia
- kaksi teknillisesti toimivaa pelisivua.

Työkalujen ja tekniikoiden valinnat olivat tekijän mielestä hyviä, koska tehdyt pelisivut saatiin toteutettua kuten suunniteltiin. Erityisesti CodeIgniterin valinta PHP-pohjaiseksi sovelluskehykseksi oli tekijän mielestä erinomainen valinta. CodeIgniterin käyttö nopeutti pelin kehittämistä huomattavasti ja selkeytti pelin rakennetta.

Pelin integroiminen Facebookiin oli varsin vaivatonta, ja Facebook-sovellusalustan tarjoamat ominaisuudet olivat helppoja käyttää ja selkeästi dokumentoituja. Facebook Developerin sivuilta löytyi myös Facebook-sovellusalustaan liittyvä keskustelufoorumi, josta löytyi hyödyllistä tietoa Facebook-sovellusalustan ominaisuuksista. Facebook-integroimisen vuoksi peliin ei tarvinnut kehittää kirjautumisjärjestelmää, mikä vähensi tekijän työmäärää.

Pelin integroiminen Facebookiin toi myös negatiivisia puolia. Pelatakseen pelaajalla täytyy olla aktiivinen Facebook-tili, mikä vähentää potentiaalisten pelaajien lukumäärää. Sovelluksen kehittäjän tulee myös aktiivisesti seurata Facebookin tekemiä muutoksia sovellusalustaansa, koska ne voivat helposti vaikuttaa sovelluk-

sen toimivuuteen. Facebook Developerin sivuilta löytyi useita tekniikoita, jotka Facebook aikoo lakkauttaa tai korvata uusilla tekniikoilla.

Pelin suunnittelun teorian tutkiminen jäi vähäiseksi, mutta tekijän mielestä riittäväksi. Pelille saatiin kehitettyä päätavoite, josta pelin kehittäminen voitiin aloittaa.

8 YHTEENVETO

Työn kehittäminen oli tekijälle erittäin mielenkiintoinen ja informatiivinen kokemus. Aihealue oli varsin laaja, ja projektin tekemiseen kulutettiin useita työtunteja. Työn toteuttaminen edellytti useiden tekniikoiden ja työkalujen hallintaa, joita opeteltiin tarvittaessa. CodeIgniterin tarkempi tutkiminen useista lähteistä valotti CodeIgniterin ominaisuuksia, joita hyödynnettiin myös pelin kehittämisessä. CodeIgniteriin tutustuminen helpottaa myös tutustumista muihin PHP-pohjaisiin sovelluskehyksiin, koska ne usein toimivat samalla periaatteella. Alla olevissa kuvissa esitellään projektin tavoiteaikataulu ja toteutunut aikataulu.

Tavoiteaikataulu								
	2011					2012		
Tehtävä	**elo**	**syys**	**loka**	**marras**	**joulu**	**tammi**	**helmi**	**maalis**
Peli-idean suunnittelu								
Pelin teknilliset päälinjaukset								
PHP-pohjaisen sovelluskehyksen valinta ja tutkiminen								
Pelipohjan kehitys								
Pelaajan rekisteröinti								
Pelin taistelusysteemin kehittäminen								
Muiden peliominaisuuksien kehittäminen								

Kuva 25. Työn tavoiteaikataulu.

Toteutunut aikataulu								
	2011					2012		
Tehtävä	**elo**	**syys**	**loka**	**marras**	**joulu**	**tammi**	**helmi**	**maalis**
Peli-idean suunnittelu								
Pelin teknilliset päälinjaukset								
PHP-pohjaisen sovelluskehyksen valinta ja tutkiminen **(tutkittiin sitä myöten kun siihen tuli tarvetta)**								
Pelipohjan kehitys								
Pelaajan rekisteröinti								
Pelin taistelusysteemin kehittäminen								
Muiden peliominaisuuksien kehittäminen **(Ei saatu valmiiksi)**								

Kuva 26. Työn toteutunut aikataulu.

LÄHTEET

CodeIgniter User Guide Version 2.1.0. 2010a. Application flow chart. [Verkkosivu]. EllisLab. [Viitattu 17.2.2012]. Saatavana: http://codeigniter.com/user_guide/overview/appflow.html.

CodeIgniter User Guide Version 2.1.0. 2010b. CodeIgniter at a Glance. [Verkkosivu]. EllisLab. [Viitattu 17.2.2012]. Saatavana: http://codeigniter.com/user_guide/overview/at_a_glance.html.

CodeIgniter User Guide Version 2.1.0. 2010c. Creating libraries.[Verkkosivu]. EllisLab. [Viitattu 17.2.2012]. Saatavana: http://codeigniter.com/user_guide/general/creating_libraries.html.

CodeIgniter User Guide Version 2.1.0. 2010d. Helper functions.[Verkkosivu]. EllisLab. [Viitattu 17.2.2012]. Saatavana: http://codeigniter.com/user_guide/general/helpers.html.

CodeIgniter User Guide Version 2.1.0. 2010e. Model-View-Controller.[Verkkosivu]. EllisLab. [Viitattu 17.2.2012]. Saatavana: http://codeigniter.com/user_guide/overview/mvc.html.

Feuerstein, S. & Pribyl, B. 2002. Oracle PL/SQL Programming, Third Edition. 3. p. Sebastopol: O'Reilly & Associates, Inc.

Gehtland, J. & Galbraith, B. & Almaer, D. 2006. Pragmatic Ajax: A Web 2.0 Primer. 2. p. Yhdysvallat.

Griffiths, A. 2010. CodeIgniter 1.7 Professional Development: Become a CodeIgniter expert with professional tools, techniques, and extended libraries. Birmingham: Packt Publishing Ltd.

Griffiths, A. 2010. CodeIgniter 1.7 Professional Development: Become a CodeIgniter expert with professional tools, techniques, and extended libraries. Birmingham: Packt Publishing Ltd.

Heinisuo, R. 2004. PHP ja MySQL: Tietokantapohjaiset verkkopalvelut. 3. uud. p. Helsinki: Talentum.

Korpela, J. 2008. CSS verkkosivujen muotoilussa. 1. p. Porvoo: WS Bookwell.

Linjama, T. 1998. HTML 4. 1. p. Jyväskylä: Teknolit.

Lobão, A. & Evangelista, B. & Farias, J. & Grootjans, R. 2009. Beginning XNA 3.0 game programming: From Novice to Professional. New York: Springer-Verlag New York, Inc.

Moncur, M. 2000. JavaScript Trainer. Helsinki: IT Press.

Parker, D. 2005. Understanding the HTTP protocol (Part 1). [Verkkosivu]. Tech-Genix Ltd. Saatavana: http://www.windowsnetworking.com/articles_tutorials/understanding-http-protocol-part1.html

Rantala, A. 2005. Web-ohjelmointi. 1. p. Porvoo: WS Bookwell.

Smith, R. 2006. What is CRON and what can it do. [Verkkosivu]. Developer Shed. Saatavana: http://webhosting.devshed.com/c/a/Web-Hosting-Articles/What-is-CRON-and-What-Can-it-Do/.

LIITTEET

LIITE 1 Facebook Ignitedin asetukset

```
<?php  if ( ! defined('BASEPATH')) exit('No direct script access allowed');
/**
 * --- Facebook Ignited ---
 *
 * fb_appid     is the app id you recieved from dev panel
 * fb_secret    is the secret you recieved from dev panel
 * fb_canvas    the value you put in 'Canvas Page' field in dev panel
                or the address of your app.
 * fb_apptype   set to either 'iframe' or 'connect' based on what
                platform your app is running on.
 * fb_auth      is the default authentications, '' is basic authentication
 */
$config['fb_appid']     = '';
$config['fb_secret']    = '';
$config['fb_canvas']    = 'http://apps.facebook.com/bunny_vs_magician/';
$config['fb_apptype']   = 'iframe';
$config['fb_auth']      = '';
```

LIITE 2 Facebook-sovellusalustan JavaScript-kirjaston latauskoodi

```
window.fbAsyncInit = function() {
   FB.init({
     appId      : '', // App ID
     status     : true, // check login status
     cookie     : true // enable cookies to allow
                       // the server to access the session
   });

   // Additional initialization code here
 };

 // Load the SDK Asynchronously
 (function(d){
    var js, id = 'facebook-jssdk'; if (d.getElementById(id)) {return;}
    js = d.createElement('script'); js.id = id; js.async = true;
    js.src = "//connect.facebook.net/en_US/all.js";
    d.getElementsByTagName('head')[0].appendChild(js);
  }(document));
```

LIITE 3 jQueryllä tehty Ajax-kutsun perusrakenne

```
$.ajax({
        type: "POST",
        url: url,
        data: "parametri="+parametri,
        success: function(result){
                //tänne tullaan jos ajax-kutsu suoritettiin onnistuneesti
        },
        beforeSend: function(){
                //tänne tullaan ennen ajax-kutsun lähetystä
        },
        complete: function(){
                //tänne tullaan kun ajax-kutsu on suoritettu
        }
});
```