HUD-EDITORI WEB-SOVELLUKSENA

Case: ToonHUD.com

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Visamäki, Tietojenkäsittelyn koulutusohjelma

Syksy, 2016

Joose Kaasalainen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutusohjelma
Visamäki

Tekijä Joose Kaasalainen **Vuosi** 2016

Työn nimi HUD-editori web-sovelluksena

TIIVISTELMÄ

Opinnäytetyön aiheena oli kehittää web-sovellus Windows-työpöytäsovelluksesta. Toteutettu sovellus oli editori Toonhudille. ToonHUD on opinnäytetyön tekijän toteuttama HUD Team Fortress 2 -peliin. Opinnäytetyöllä ei ollut erillistä toimeksiantajaa, vaan se toteutettiin henkilökohtaisena projektina.

Opinnäytetyön teoriaosuudessa avataan ensin projektin lähtökohdat: mikä on Team Fortress 2, HUD ja ToonHUD. Sen jälkeen kerrotaan työpöytäsovelluksen ominaisuudet ja syyt miksi sovellus haluttiin toteuttaa uudelleen web-sovelluksena. Lopuksi siinä käydään läpi web-sovelluksen tekoon käytetyt tekniikat ja kirjastot esimerkkien kera.

Opinnäytetyön käytännönosuudessa keskitytään web-sovelluksen kahteen tärkeimpään ominaisuuteen: käyttäjien tunnistautuminen sovellukseen Steam-palvelun kautta sekä sovelluksella tehtävien teemojen hallinta. Teemojen hallinnasta käydään läpi eri toimintojen logiikka: minkälaisia asioita tehdään, missä järjestyksessä ja miksi.

Tekijällä oli ennestään kokemusta web-sivustojen ohjelmoinnista. Uusina asioina opittiin kuitenkin mm. sovellusmaisen sivuston teko, kirjautuminen kolmannen osapuolen palvelun kautta sekä erilaisten JavaScript-kirjastojen käyttö.

Työn lopputuloksena saatiin toimiva ja tavoitteet täyttävä web-sovellus, jolla on jo kymmeniä tuhansia rekisteröityneitä käyttäjiä. Työpöytäsovellukseen verrattuna web-sovelluksen ylläpito todettiin helpommaksi ja mielekkäämmäksi, vaikka työpöytäsovelluksessa oli myös omat puolensa.

Avainsanat web-sovelluksen kehitys, HUD, JavaScript, PHP

Sivut 39 sivua, joista liitteitä 5 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Degree Programme in Business Information Technology
Visamäki

Author Joose Kaasalainen **Year** 2016

Subject HUD Editor as a Web Application

ABSTRACT

The goal of this thesis was to recreate a Windows desktop application as a web application. The application that was recreated was an editor for ToonHUD. ToonHUD is a custom HUD for a game called Team Fortress 2 that was created by the author of this thesis. This project did not have a client as it was a personal project of the author.

The theoretical section of this thesis explains the background information that is needed to understand the concept of this work: what is Team Fortress 2, HUD and ToonHUD. Then it explains the features of the desktop application and reasoning why it was chosen to be replaced with a web application. Lastly it goes through the techniques and JavaScript libraries that were used to make the web application. It also provides small code examples of each technique.

The practical section focuses on the two main features of the web application: user authentication through Steam and theme management. The theme management part explains the logic of each action: what is done in which order and why.

Before writing this thesis, the author had some previous knowledge on web development. New things that were learned during the project was how to create application-like websites and how to add third-party authentication to them. Usage of a few new JavaScript libraries was learned as well.

The result of this project is a fully functional web application which fulfilled the given requirements. It has already tens of thousands of registered users. Compared to the desktop application, the web application was considered easier and more pleasant to maintain, even though the desktop application had its own great features as well.

Keywords web application development, HUD, JavaScript, PHP

Pages 39 pages including appendices 5 pages

KÄSITELUETTELO

AJAX (Asynchronous JavaScript And XML)
Tekniikka, joka mahdollistaa HTTP-pyyntöjen suorittamisen ja niiden tulosten palauttamisen käyttäjälle taustalla, web-sivua päivittämättä.

C# (C Sharp)
Microsoftin kehittämä ohjelmointikieli, pääasiassa Windows-alustoille.

CSS (Cascading Style Sheets)
Web-sivujen tyylittelyyn suunniteltu kieli.

DOM (Document Object Model)
Web-sivuilla DOM kuvaa sivuston elementit puumaisena rakenteena. Selaimet luovat DOM-rakenteen sivulatauksen yhteydessä. DOM-rakennetta voidaan manipuloida JavaScriptillä.

Eväste (Cookie)
Tietoa, jonka web-palvelin tallentaa käyttäjän selaimeen. Eväste lähetetään takaisin palvelimelle jokaisella sivulatauksella.

Funktio (Function)
Aliohjelma sovelluksessa, joka suorittaa jonkin tietyn toiminnon. Funktioita voidaan kutsua eri puolilta sovellusta. Funktiolle voidaan syöttää tietoa käsiteltäväksi parametrien avulla.

HTML (HyperText Markup Language)
Web-sivujen tekoon suunniteltu merkkauskieli.

HUD (Heads-Up Display)
Tietoa näyttävä taso, jonka käyttäjä näkee kääntämättään katsettaan pääkohteesta. Hudia käytetään mm. sotalentokoneissa, autojen navigaatiojärjestelmissä sekä videopeleissä.

JavaScript
Internet-selaimessa suoritettava ohjelmointikieli, joka mahdollistaa interaktiivisten web-sivustojen teon.

JavaScript-kirjasto (JavaScript library)
JavaScript-kirjastot sisältävät JavaScript-kielellä kirjoitettuja valmiita toimintoja, joita kehittäjä voi käyttää yksinkertaistaakseen ja nopeuttaakseen JavaScript-ohjelmointiaan.

jQuery
JavaScript-kirjasto, joka mahdollistaa JavaScript-koodin kirjoittamisen lyhyemmin ja yksinkertaisemmin.

Microsoft Visual Studio
Helppokäyttöinen integroitu ohjelmointiympäristö (IDE) Windows-työpöytäsovellusten sekä web-sivujen ja -sovellusten toteutukseen.

MySQL
Oracle Corporationin kehittämä, maailman suosituin SQL-tietokantajärjestelmä.

Ohjelmointirajapinta (API, Application Programming Interface)
Väylä jota pitkin eri sovellukset voivat jakaa tietoansa toisille sovelluksille.

PHP (Hypertext Processor)
Web-kehitykseen suunniteltu avoimen lähdekoodin ohjelmointikieli.

Steam
Valve Corporationin kehittämä alusta, jonka avulla pelaajat voivat ostaa ja ladata pelejä sekä kommunikoida keskenään.

SQL (Structured Query Language)
Oletuskieli tietokantojen hallintaan.

Team Fortress 2 (TF2)
Valve Corporationin kehittämä ensimmäisen persoonan ammuntapeli.

ToonHUD
Opinnäytetyön kirjoittajan kehittämä muokattu versio Team Fortress 2 -pelin hudista.

XML (Extensible Markup Language)
Merkkauskieli jäsennellyn tiedon säilytykseen.

ZIP
Tiedonpakkausmenetelmä. Tiedostot saadaan pienempään tilaan pakkaamalla ne ZIP-paketeiksi. ZIP-pakettien tiedostopääte on “.zip”.

SISÄLLYS

Liitteet

Liite 1 Vuokaavio uuden teeman luomisesta
Liite 2 Vuokaavio teeman tallennuksesta
Liite 3 Vuokaavio teeman muokkauksesta
Liite 4 Vuokaavio teeman kopioinnista/monistuksesta
Liite 5 Vuokaavio teeman latauksesta

1 JOHDANTO

Nopeat internetyhteydet ja tehokkaat päätelaitteet ovat mahdollistaneet monipuolisten web-sovellusten kehityksen. Monet sovellukset toimivat nykyään suoraan selaimella ilman että käyttäjän tulee ladata erillistä sovellusta päätelaitteelleen. Sovelluksissa tuotettu datakin voidaan tallentaa suoraan pilveen. Kaikki on internet-yhteyden päässä, päätelaitteesta riippumatta.

Opinnäytetyön aiheena on toteuttaa web-sovellus aiemmin toteutetusta C#-pohjaisesta Windows-työpöytäsovelluksesta. Toteutettava sovellus on editori Toonhudin muokkaukseen. ToonHUD on opinnäytetyön kirjoittajan kehittämä muokattu versio Team Fortress 2 PC-pelin hudista. Aiempaa sovellusta ei siirretä uuteen alustaan, vaan se toteutetaan uudelleen eri tekniikoilla: pääasiassa PHP:lla ja erilaisilla JavaScript-kirjastoilla.

Projekti lähti liikkeelle halusta oppia web-tekniikoita ja parantaa Toonhudin käyttäjien käyttökokemusta. Opinnäytetyöllä ei ole erillistä tilaajaa, vaan aihe on opinnäytetyön kirjoittajan henkilökohtainen projekti.

Työn lopputuloksena on tarkoitus saada toimiva web-sovellus, johon käyttäjät pystyvät kirjautumaan Steam-tunnuksillaan. Sovelluksen avulla käyttäjien tulisi voida muokata Toonhudia mieleisekseen, sekä ladata ja jakaa muokkauksiaan, joita kutsutaan teemoiksi.

Opinnäytetyön teoriaosuudessa avataan ensin projektin lähtökohdat: mikä on Team Fortress 2, HUD ja ToonHUD. Sen jälkeen käydään läpi alkuperäisen sovelluksen ominaisuudet ja syyt uuden sovelluksen toteutukseen. Lopuksi selitetään uudessa web-sovelluksessa käytetyt tekniikat.

Opinnäytetyön käytännön osuudessa keskitytään sovelluksen toteutukseen ja sen tärkeimpiin ominaisuuksiin: käyttäjien tunnistautuminen ja teemojen hallinta.

Opinnäytetyö vastaa muun muassa seuraaviin kysymyksiin:

- Mitkä ovat web-sovelluksen hyödyt ja haitat aiempaan työpöytäsovellukseen verrattuna?
- Miten toteutetaan tiedostoja muokkaava web-sovellus mainituilla tekniikoilla?
- Miten web-sivustolle saa lisättyä kirjautumisen Steam-palvelun kautta?

2 PROJEKTIN LÄHTÖKOHDAT

Opinnäytetyön ymmärtämisen kannalta on tiedettävä hieman projektin lähtökohdista. Tässä luvussa kerrotaan oleellisimmat asiat opinnäytetyön lähtökohdista ja projektin historiasta: mikä on Team Fortress 2, HUD ja ToonHUD.

2.1 Team Fortress 2

Team Fortress 2, lyhennettynä TF2, on Valve Corporationin kehittämä huumoristaan ja sarjakuvamaisesta grafiikastaan tunnettu ensimmäisen persoonan ammuntapeli. Peli julkaistiin lokakuussa 2007 ostettavaksi kauppoihin The Orange Box -nimisen pelipaketin mukana, sekä Steamin verkkokauppaan itsenäisenä pelinä. Kesäkuussa 2011 Valve teki pelistä ilmaiseksi pelattavan. Tuottoa pelistä Valve saa kuitenkin pelin sisäisten ostojen kautta. Peliin voi ostaa mm. aseita, hattuja ja muita kosmeettisia esineitä. (Official TF2 Wiki 2016a.)

Pelissä sotivat vastakkain kaksi joukkuetta: punainen joukkue RED (Reliable Excavation Demolition) ja sininen joukkue BLU (Builders League United). Pelin taustatarinan mukaan, sininen joukkue on rakennusyritys, kun taas punainen joukkue on purkuyritys. (Official TF2 Wiki 2016a.)

Pelissä on useita pelimuotoja, joista suosituimmat ovat Pommilasti, Valtauspisteet, Kukkulan kuningas sekä Lipunryöstö. Pommilastissa sinisen joukkueen tehtävä on työntää pommilasti raiteita pitkin kartan päähän annetussa ajassa, kun punaisen joukkueen tehtävä on estää se. Valtauspisteet-pelimuoto vaihtelee kartasta riippuen. Yleensä molemmat joukkueet yrittävät vallata kaikki valtauspisteet. Joissain kartoissa toinen joukkue yrittää vallata ja toinen estää valtauksen. Kukkulan kuninkaassa on yksi valtauspiste, jota molemmat joukkueet yrittävät vallata. Se joukkue voittaa, kumpi on pitänyt pisteen vallattuna ilmoitetun ajan. Lipunryöstössä molemmat joukkueet yrittävät varastaa toistensa salaista tietoa sisältävän salkun. (Official TF2 Wiki 2016b.)

Pelissä on yhdeksän pelattavaa hahmoluokkaa: Scout, Soldier, Pyro, Demoman, Heavy, Engineer, Medic, Sniper ja Spy. Kaikilla hahmoluokilla on oma roolinsa pelissä. Hahmoluokat poikkeavat toisistaan eri asevalikoimillaan, liikkumisnopeuksillaan ja terveyspisteillään. Koska hahmoluokat poikkeavat pelattavuudeltaan toisistaan, peli pysyy mielenkiintoisena pitkään.

Pelillä on laaja pelaajakanta. Pelin ilmaisuus ja kepeä ulkoasu tekevät siitä helposti lähestyttävän. Steam tarjoaa tilaston, joka näyttää suosituimmat pelit sen hetkisen pelaajamäärän mukaan (Steam 2016a). Sitä seuraamalla selviää, että Team Fortress 2 pysyy listan kärkipäässä.

Suurin osa pelissä käytetyistä kartoista, aseista ja kosmeettisista esineistä on yhteisön tekemiä. 3D-mallinnuksen taitavat kehittäjät voivat luoda esineitä peliin. Kun esine on valmis, kehittäjä voi lähettää sen Steam Workshopiin. Steam Workshop on paikka, jonne kehittäjät voivat lähettää tekemiään töitä muiden pelaajien ladattavaksi ja arvosteltavaksi. Jos esine täyttää Valven asettamat kriteerit, esineellä on mahdollisuus päätyä peliin. Jos esine lisätään peliin, se lisätään myös myyntiin TF2:n omaan sisäiseen esinekauppaan, Mann Co. Storeen. Esineen kehittäjä saa 25 prosenttia esineen kaupassa tuottamista tuloista. (Steam n.d.b.)

2.2 **HUD**

HUD on lyhenne sanoista Heads-Up Display, josta käytetään suomenkielessä sanaa heijastusnäyttö. HUD tarkoittaa informatiivista näyttöä, joka antaa käyttäjälle tietoa suoraan silmien eteen niin, ettei käyttäjän tarvitse kääntää katsettaan lukeakseen tietoa. Ensimmäisiä heijastusnäyttöjä käytettiin sotilasajoneuvoissa ja -lentokoneissa navigointiin ja tähtäykseen. Nykyään hudeja myydään erilaisina laitteina, kuten auton navigaattoreina sekä älylaseina, kuten Google Glass. (Margaret Rouse 2013.)

Termiä HUD käytetään myös videopeleissä. HUD on erillinen taso pelin päällä, joka kertoo pelaajalle oleellista tietoa pelin tapahtumista. Hudissa näkyvät tiedot vaihtelevat peleittäin. Esimerkiksi autopeleissä tyypillisiä HUD-elementtejä ovat nopeusmittarit, kierrosten lukumäärä ja kierrosajat. Sotapelien hudeista löytyy yleensä pelaajan terveyspisteet, ammusten määrä, kartta ja kompassi.

Kuva 1. Team Fortress 2 -pelin HUD korostettuna.

Kuva 2. Super Mario Bros. ja Final Fantasy VIII -pelien hudit korostettuina.

Hyvä HUD on intuitiivinen. Hudin päätavoite on antaa pelaajalle tietoa nopeasti ja tehokkaasti niin, ettei pelaajan tarvitse nähdä vaivaa etsiäkseen oleellista tietoa. Katseen tulisi kääntyä automaattisesti sillä hetkellä tärkeimpään asiaan. HUD ei myöskään saisi peittää peliä liikaa liian suurella tietomäärällä tai suurilla elementeillä. Hudin tulisi olla mahdollisimman huomaamaton silloin kun sitä ei käytetä. (Melissa Loomis 2015.)

Hudia ei tule sotkea pelin käyttöliittymään. HUD avustaa pelaajaa näyttämällä tietoa pelistä pelaamisen aikana. Käyttöliittymä sisältää kaiken interaktiivisen, kuten valikot ja muut valinnat pelissä. (Mark Masters 2013.) TF2-peliin tehdyt käyttäjien muokkaamat hudit sisältävät usein sekä käyttöliittymän että hudin muokkauksia, koska ne rakentuvat samanlaisista tiedostoista. Tästä huolimatta niitä kutsutaan yleensä vain hudeiksi.

Team Fortress 2 -aiheisia keskustelupalstoja seuranneena, olen huomannut, että TF2:n kilpapelaajat, tai muuten vakavammin pelin ottavat pelaajat, käyttävät usein muokattuja hudeja. TF2 on nopeatempoinen peli, jossa katse ei aina ehdi herpaantua pelistä esimerkiksi terveyspisteiden lukemiseen. TF2:n oma HUD ei ole parhaimmasta päästä. HUD-elementit ovat ruudun laidoissa ja ne ovat isoja, mutta itse tieto niissä on suhteellisen pienellä. Tämä sopii kasuaaliin pelaamiseen, mutta kilpapelaamiseen se ei välttämättä ole riittävän intuitiivinen. Kilpapelaajat suosivat yleensä hudeja, joissa tieto on tuotu lähelle tähtäintä, johon katse pääasiassa kiinnittyy pelin aikana.

2.3 **ToonHUD**

ToonHUD on kehittämäni HUD Team Fortress 2 peliin. Ajatus hudin kehittämiseen lähti mielenkiinnosta sekä itse peliä, että ohjelmointia kohtaan. Peliin oli tehty jo ennestään kustomoituja hudeja. Niistä inspiroituneena halusin tehdä oman version.

ToonHUD sai nimensä käyttämästään sarjakuvamaisesta fontista. Sana “toon” on lyhenne englanninkielisestä sanasta “cartoon”, suomeksi sarjakuva.

Kuva 3. ToonHUD.

Team Fortress 2 -pelin HUD koostuu pääasiassa res-tiedostoista (Resource) sekä vtf-tiedostoista (Valve Texture Format). RES-tiedostot sisältävät kaikki pelin käyttöliittymän ja hudin elementit ja niiden tyylit. RES-tiedostot muistuttavat merkintäkieleltään CSS-tyylittelyä. VTF-tiedostot ovat tekstuuritiedostoja, eli käytännössä kuvatiedostoja. ToonHUD toteutettiin muokkaamalla näitä tiedostoja. Muokatut tiedostot pistettiin omaan toonhud-nimiseen kansioon pelin oman hudin tiedostorakenne säilyttäen. Toonhudia käytetään siten, että toonhud-kansio siirretään pelin asennuskansiossa olevan custom-kansion sisään. Kun peli käynnistetään, ToonHUD on käytössä.

Heinäkuussa 2013, kun ToonHUD oli pääpiirteittäin valmis, julkaisin sen muiden ladattavaksi GameBanana.com-sivustolle. GameBanana on sivusto, jonne käyttäjät voivat ladata tekemiään muokkauksia eri peleihin. ToonHUD sai sivuston kommenttiosiossa pääosin hyvää palautetta. Eniten keskustelua nostatti rohkea sarjakuvamainen fontti. Monet eivät siitä pitäneet, kun taas osa piti sitä juuri sopivana TF2:n sarjakuvamaiseen tyyliin. Kommenteissa annettiin myös paljon kehitysehdotuksia ja virheraportteja. Käyttäjien mielenkiinto Toonhudia kohtaan sai minut jatkamaan sen ylläpitoa.

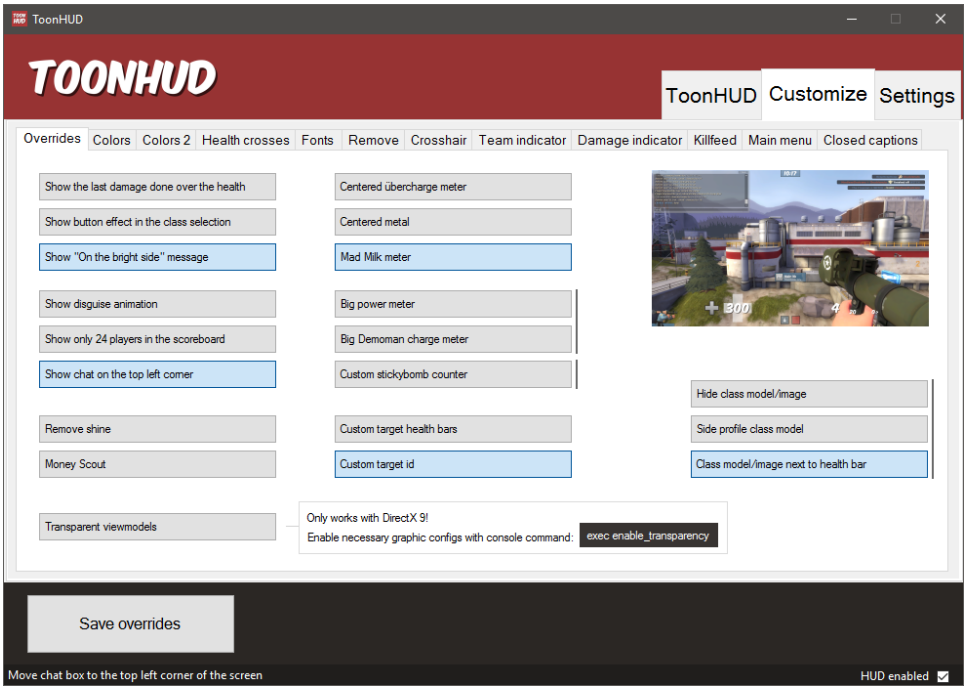
Myöhemmin tein Toonhudille omat kotisivut, josta käyttäjät pystyivät lataamaan hudin, jättämään palautetta sekä näkemään hudin versiohistorian.

3 ALKUPERÄINEN SOVELLUS

Käyttäjillä on monia mielipiteitä siitä, miltä hudin tulisi näyttää. Toiset pitävät esimerkiksi tietystä fontista tai väristä, kun toiset taas eivät. Halusin antaa käyttäjille mahdollisuuden muokata valittuja Toonhudin ominaisuuksia ilman, että heidän tarvitsisi tietää hudien ohjelmoinnista mitään. Päätin tehdä Toonhudille oman C#-pohjaisen työpöytäsovelluksen Windows-käyttöjärjestelmälle. Toteutin sovelluksen koulussa tutustumallani kehitysympäristöllä: Microsoft Visual Studiolla. Sovelluksen nimeksi tuli ToonHUD Updater, suomeksi ToonHUD-päivittäjä. Sovellus julkaistiin huhtikuussa 2014.

ToonHUD Updaterilla käyttäjät pystyivät muokkaamaan Toonhudia mieleisekseen. Käyttäjät pystyivät vaihtamaan fontteja sekä eri tekstien ja valikoiden värejä. Sovelluksessa oli myös monia vaihtoehtoisia ominaisuuksia, jotka toimivat päälle/pois-periaatteella. Näihin kuului muun muassa eri muotoiset ja kokoiset tähtäimet, terveyspiste- ja voimamittarit sekä pistetaulukot. Monet ominaisuudet toteutin käyttäjien pyynnöstä.

ToonHUD Updaterin kenties paras ominaisuus oli sen helppokäyttöinen päivitysominaisuus. Käyttäjät pystyivät napin painalluksella lataamaan uusimman version Toonhudista suoraan koneelleen, vieläpä oikeaan kansioon pelin sisään. Kun uusin versio oli ladattu, sovellus lisäsi käyttäjän tekemät muokkaukset hudiin automaattisesti.

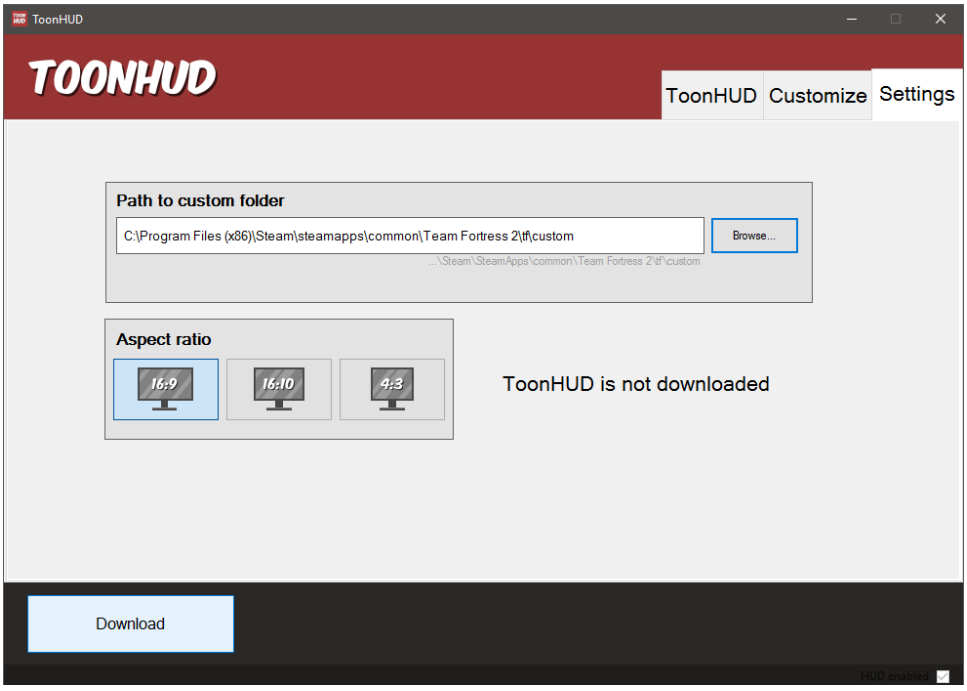


Kuva 4. Alkuperäinen sovellus - ToonHUD Updater.

3.1 Sovelluksen toiminta

Sovellus koostui kahdesta sovellustiedostosta: HUD-editorin päivittäjästä ja itse HUD-editorista. HUD-editorin päivittäjän tiedostonimi oli ”ToonHUD Updater.exe” ja itse HUD-editorin ”thu.exe”. Kun HUD-editorin päivittäjä käynnistettiin, se tarkisti toonhud.com-sivustolla olevasta tekstitiedostosta HUD-editorin versionumeron. Sama tekstitiedosto sisälsi myös URL-osoitteen, mistä HUD-editorin päivitys ladattaisiin. Tiedosto sisälsi versionumeron ja latausosoitteen myös Toonhudille. Jos tekstitiedostossa ollut versionumero oli suurempi kuin sovelluksessa oleva, päivitys ladattiin tekstitiedostossa olleesta osoitteesta. Päivitystiedosto oli ZIP-paketti, joka sisälsi päivitetyn thu.exe-tiedoston. Tiedosto ladattiin sovelluskansion sisään. Sen jälkeen vanha thu.exe-tiedosto poistettiin ja uuden thu.exe-tiedoston sisältänyt ZIP-paketti purettiin, korvaten näin vanhan sovelluksen. Kun päivitys oli valmis, HUD-editorin päivittäjä käynnisti päivitetyn HUD-editorin.

HUD-editorin käynnistyessä ensimmäisen kerran, se pyysi käyttäjää hakemaan polun Team Fortress 2 -pelin custom-kansioon, johon kaikki pelin muokkaukset sijoitetaan. Näin sovellus tietäisi mihin ladattu HUD tulisi pistää. Sovellus kysyi myös käyttäjän tietokoneen näytön kuvasuhdetta. ToonHUD on pääasiassa suunniteltu laajakuvanäytöille, mutta muutamalla tiedostomuokkauksella sen saa toimimaan myös 4:3-näytöillä. Sovellus teki nämä muokkaukset, jos käyttäjä valitsi kuvasuhteekseen 4:3. Kun asetukset oli määritelty, Download-painikkeella asetukset tallentuivat ja Toonhudin lataus alkoi.



Kuva 5. ToonHUD Updater -sovelluksen Asetukset-välilehti.

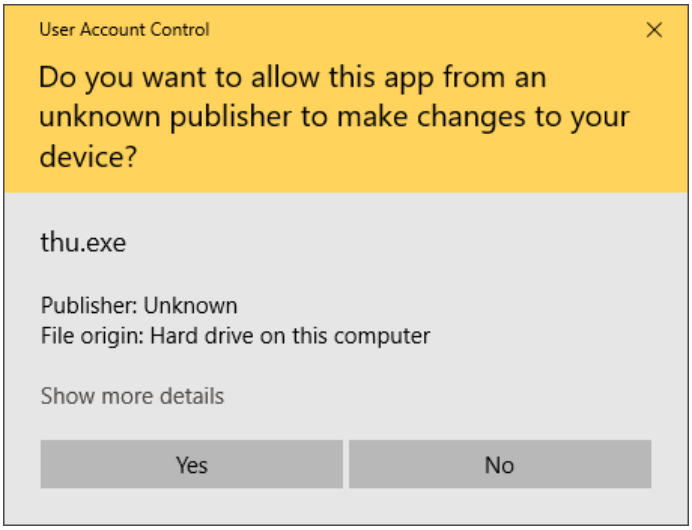
ToonHUD ZIP-paketti ladattiin sovelluskansiossa olevaan data-kansioon, jonka jälkeen ZIP-paketti purettiin. Näin data-kansiossa oli aina puhdas, muokkaamaton ToonHUD.

Kun HUD-editori käynnistettiin, se tarkasti aina, onko Toonhudista uutta versiota saatavilla. Jos päivitys löytyi, sovelluksen etusivulle tuli tieto tästä. Käyttäjä pystyi päivittämään Toonhudin napin painalluksella. Päivitys ladattiin samalla tapaa kuin ensimmäisellä kerralla. Data-kansiossa ollut toonhud-kansio poistettiin ennen päivitetyn version lisäämistä.

Jotta HUD saataisiin näkymään pelissä, se tulisi siirtää pelin custom-kansioon. Tämä tapahtui aina päivityksen jälkeen sekä silloin, kun käyttäjä teki muutoksia hudiin editorin kautta, esimerkiksi vaihtoi väriä tai fonttia. Sovellus poisti pelin custom-kansiossa olevan toonhud-kansion, jos sellainen oli olemassa. Sen jälkeen sovellus kopioi data-kansiossa olevan muokkaamattoman toonhud-kansion custom-kansion sisään ja teki tarvittavat muokkaukset hudiin.

3.2 Sovelluksen ongelmat

ToonHUD Updater oli pidetty uudistus - ainakin niiden keskuudessa kenellä kyseinen sovellus toimi. Koska ToonHUD Updater oli itse tehty sovellus, sillä ei ollut virallista julkaisijaa. Windows varoitti tästä aina kun sovelluksen käynnisti. Osa virustorjuntaohjelmista jopa poisti sovelluksen tietokoneelta automaattisesti heti kun sen käynnisti, koska ne eivät tienneet voiko sovellukseen luottaa. Jotta sovelluksesta saisi luotettavan myös virustorjuntaohjelmien keskuudessa, sovellus vaatisi maksullisen koodin allekirjoitusvarmenteen. Koodin allekirjoitusvarmenteella taataan se, että sovellus on tullut kehittäjältä, eikä sitä ole kukaan kolmas taho muokannut. En ollut valmis maksamaan tästä noin kahtasataa euroa vuodessa.



Kuva 6. Windowsin varoitus tuntemattomasta julkaisijasta.

Sovelluksen tuntematon julkaisija ei välttämättä ollut ainoa ongelma virustorjuntaohjelmien keskuudessa. ToonHUD Updater oli ensimmäinen työpöytäsovellus, jonka olin tehnyt. Sovelluksen koodissa saattoi olla aukkoja ja tietoturvariskejä, joista en vain ollut tietoinen. Kokemusta työpöytäsovellusten tekemisestä ei vaan ollut tarpeeksi.

Tuntemattomien sovellusten asennus tietokoneelle sisältää aina riskin. Siksi kaikki käyttäjät eivät uskaltaneet ladata ToonHUD Updateria koneelleen.

ToonHUD Updater oli kehitetty vain Windowsille. Macin OS X:n käyttäjät ja Linux-käyttäjät eivät voineet käyttää sovellusta.

Mainitut ongelmat saivat minut harkitsemaan uudenlaisen sovelluksen toteutusta. Kun löysin tavan web-sovelluksen toteutukseen, lähdin toteuttamaan sitä.

4 WEB-SOVELLUS

Web-sivut ovat yleensä sivuja, jotka sisältävät staattista, harvemmin muuttuvaa sisältöä, jonka käyttäjä avaa vain lukeakseen. Web-sovellukset poikkeavat tavallisista web-sivustoista antamalla käyttäjälle mahdollisuuden tehdä jotain muutakin kuin vain lukea sen sisältöä. Web-sovelluksessa käsitellään dataa ja se on interaktiivinen käyttäjän kanssa. Raja ei ole kuitenkaan aina niin selvä. Onko yrityksen kotisivut web-sovellus, jos se sisältää yhteydenottolomakkeen? Entä sivustot, jotka on rakennettu sisällönhallintajärjestelmän päälle? (Philip Rostamadji 2015.)

Web-sovellus tehtiin korvaamaan aiempi Windows-sovellus. Uuteen sovellukseen haluttiin samat muokkausmahdollisuudet, mitä Windows-sovellus sisälsi. Lisäksi käyttäjien tulisi pystyä kirjautumaan sovellukseen Steam-tunnuksellaan sekä jakamaan teemoja suoran linkin kautta.

4.1 Käytetyt tekniikat

Tässä luvussa kerrotaan web-sovellukseen käytettyjen tekniikoiden perusteet pienten esimerkkien kera.

4.1.1 HTML

HTML, eli HyperText Markup Language, on merkkauskieli, jota käytetään web-sivustojen rakentamiseen. HTML-koodi koostuu erilaisista elementeistä, jotka muodostavat sivuston rakenteen. HTML-elementtejä ovat esimerkiksi otsikot, linkit, tekstikappaleet, kuvat, listat, taulukot ja sisältöalueet. HTML-koodi muodostuu elementtien aloitus- ja lopetustageista, tagien attribuuteista sekä tagien väliin tulevasta sisällöstä.

HTML-tagit kirjoitetaan < ja > -merkkien sisään. Lopetustagi sisältää kauttaviivan ennen tagin nimeä. Esimerkiksi <html> on aloitustagi ja </html> on lopetustagi. Tagien avulla HTML-merkkaukselle saadaan hierarkkinen rakenne. Jokainen elementti voi sisältää useamman elementin sisällään.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sivuston otsikko</title>
        <link rel="stylesheet" type="text/css" href="styles.css" />
    </head>
    <body>
        <h2>Tämä on otsikko</h2>
        <p>Tämä on tekstikappale.</p>
    </body>
</html>
```

Kuva 7. Esimerkki yksinkertaisesta HTML-dokumentista.

Kuvan 7 esimerkkikoodi alkaa DOCTYPE-määritteellä, joka kertoo selaimelle, mitä HTML-merkkauksen versiota käytetään. Esimerkkikoodissa käytetään HTML5-versiota. DOCTYPE ei ole HTML-tagi.

Html-tagin sisään tulee kaikki HTML-koodi. Siksi sivusto yleensä alkaa ja päättyy html-tagiin.

Head-tagin sisään tulee sivun yleiset määritteet, kuten esimerkiksi selaimessa näytettävä otsikko ja sivulla käytettävät JavaScript-tiedostot sekä CSS-tyylitiedostot.

Title-tagia käytetään head-tagin sisällä. Se kertoo sivun otsikon, joka näytetään selaimen otsikkopalkissa ja välilehdissä.

Link-tagin avulla sivulle voidaan ladata sisältöä ulkopuolisista lähteistä. Link-tagia käytetään pääasiassa tyylitiedostojen ja sivuston ikonien latauksseen. Link-tagi sisältää pakollisia attribuutteja. Rel-attribuutti kertoo sivun ja ulkopuolisen tiedoston suhteen. Tässä tapauksessa suhde on stylesheet, eli tyylitiedosto. Type-attribuutti kertoo ulkopuolisen sisällön tyypin. Ensin kerrotaan tiedoston formaatti, esimerkiksi kuva, video tai teksti, jonka jälkeen kerrotaan kauttaviivalla eroteltuna tiedostopääte. Tyylitiedosto on siis text/css ja esimerkiksi PNG-kuva olisi image/png. Href-attribuutti kertoo ulkopuolisen tiedoston polun tai URL-osoitteen mistä tiedosto ladataan.

Body-tagi sisältää sivun näkyvän osuuden. Sen sisään lisätään sivulla näytettävät elementit. Kuvan 7 esimerkkikoodi sisältää vain otsikon h2-tagin sisällä sekä tekstikappaleen p-tagin sisällä.

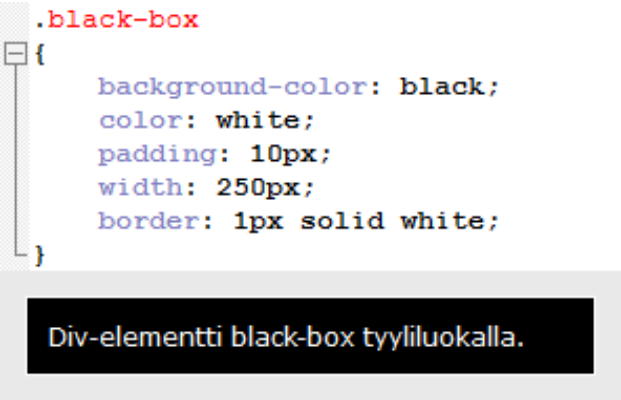
4.1.2 CSS

Pelkällä HTML-merkkauksella toteutettu web-sivu näyttää ankealta, koska sitä ei ole tyylitelty. HTML-merkkaus kertoo sivuston rakenteen, kun taas CSS-tyylitiedosto kertoo rakenteen ulkoasun. CSS-määritysten avulla HTML-elementeille kerrotaan miltä niiden tulisi näyttää. Tällaisia ominaisuuksia ovat esimerkiksi leveys, korkeus, värit, kirjaisimet, reunukset, taustakuvat jne.

CSS tulee sanoista Cascading Style Sheet. CSS-tyylimäärittelyt ladataan yleensä ulkopuolisesta tyylitiedostosta käyttäen link-tagia sivun head-osiossa. Tällöin tyylejä on helpompi hallita, ja samoja tyylejä voidaan käyttää useammalla sivulla. Tyylit voidaan kirjoittaa myös style-tagien sisässä sivun head-osioon, tai suoraan yksittäisiin elementteihin style-attribuutin avulla.

Tyylimäärittelyt koostuvat selektorista ja sen sisällä olevista säännöistä. Kaikki selektorin säännöt tulevat aaltosulkujen sisään. Jos tyylimäärittelyä halutaan käyttää kaikissa saman tyyppisissä elementeissä, voidaan selek-

toriksi valita myös elementti. Jos kaikissa tekstikappaleissa halutaan esimerkiksi käyttää samoja marginaaleja, selektori voi olla vain “p”. Selektori voi olla myös elementissä määritelty attribuutti id (tunniste) tai class (luokka). Tunnisteet ovat yksilöllisiä, joten sama tunniste saa esiintyä vain kerran HTML-dokumentissa, kun taas luokkia voi esiintyä useita samassa dokumentissa. Yksittäisellä HTML-elementillä voi olla yksi tunniste ja useita luokkia. CSS-tiedostossa tunniste-selektorit aloitetaan risuaidalla, esimerkiksi “#header”. Luokat aloitetaan pisteellä, esimerkiksi “.comment”.



Kuva 8. Esimerkki CSS-määrittelystä ja sen lopputuloksesta.

Kuvan 8 esimerkkikoodissa kirjoitettiin black-box-nimiselle luokalle tyylimäärittelyt. Taustaväri (background-color) on musta. Tekstin väri (color) on valkoinen. Ilmaa alueen sisäreunoissa (padding) on 10 pikselin verran. Alueen leveys (width) on 250 pikseliä. Alueella on yhden pikselin levyiset valkoiset reunukset (border).

4.1.3 PHP

PHP on laajasti käytetty avoimen lähdekoodin ohjelmointikieli, joka on suunniteltu erityisesti web-kehitykseen. PHP-koodia voidaan upottaa HTML-merkkauksen sekaan. (PHP.net n.d.a.)

PHP-koodia käytetään pääasiassa php-päätteisissä tiedostoissa. Muita mahdollisia tiedostopäätteitä on muun muassa php3 ja phtml. PHP-koodi alkaa aina tagilla <?php ja päättyy tagiin ?>.

PHP-koodi suoritetaan palvelimella, toisin kuin esimerkiksi JavaScript, joka suoritetaan käyttäjän selaimessa. Koska koodi suoritetaan palvelinpuolella, sivuston kävijä ei näe koodia, vaan ainoastaan sen palauttaman tuloksen. PHP-koodi suoritetaan sivun latauksen aikana. Tämä tarkoittaa sitä, että käyttäjän tarvitsee ladata sivu uudelleen, jos hän haluaa ajaa PHP-koodin uudelleen. PHP-koodia on mahdollista ajaa samalla sivulla useaan kertaan hyödyntämällä JavaScriptiä ja AJAX-kyselyitä.

```php
<?php

    // Lisää kellonaika muuttujaan $kellonaika (esim. 23:59)
    $kellonaika = date('H:i');

    // Tulosta teksti "Kello on 23:59"
    echo "Kello on ".$kellonaika;

?>
```

Kuva 9. Esimerkki PHP-koodista, joka tulostaa kellonajan tekstin sisällä.

Tässä projektissa PHP:ta käytettiin web-sovelluksen pohjana. Sivuston kaikki HTML-sivut käyttävät php-päätettä. PHP:ta käytetään mm. sisällön lataamiseen ulkoasupohjaan, keskusteluun tietokannan kanssa sekä käyttäjien kirjautumiseen.

4.1.4 XML

XML, eli Extensible Markup Language, on merkkauskieli, kuten HTML-kieli. Kun HTML-kielessä tagit ovat ennalta määrättyjä, XML-kielessä tagit voidaan nimetä tarkoituksen mukaisiksi, kuvaamaan sen sisältöä. Kun HTML keskittyy tiedon näyttämiseen, XML keskittyy tiedon säilytykseen ja lähettämiseen (W3Schools.com n.d.). XML:n avulla suuriakin tietomääriä voidaan tallentaa helposti luettavissa olevaan hierarkkiseen rakenteeseen.

Tässä projektissa XML-kieltä käytetään teemojen tietojen säilytykseen. Oletusteeman arvot sekä eri muokkaustavat luetaan XML-tiedostosta. Lisäksi muokatut teemat tallennetaan XML-muodossa tietokantaan.

```xml
<tilaukset>
    <tilaus nro="1001">
        <asiakas>
            <nimi>Joose Kaasalainen</nimi>
            <osoite>
                <katuosoite>Esimerkkikatu 3</katuosoite>
                <postinumero>13210</postinumero>
                <kaupunki>Hämeenlinna</kaupunki>
                <maa>Suomi</maa>
            </osoite>
        </asiakas>
        <maksutapa>VISA</maksutapa>
        <toimitustapa>Pakettiautomaatti S-Market Idänpää Hämeenlinna</toimitustapa>
        <pelit>
            <peli>
                <nimi>Portal 2</nimi>
                <kehittäjä>Valve</kehittäjä>
                <hinta valuutta="euro">19,90</hinta>
            </peli>
            <peli>
                <nimi>Overwatch</nimi>
                <kehittäjä>Blizzard</kehittäjä>
                <hinta valuutta="euro">60,00</hinta>
            </peli>
        </pelit>
    </tilaus>
</tilaukset>
```

Kuva 10. Esimerkki pelikaupan tilauksista XML-muodossa.

4.1.5 JavaScript

JavaScript on ohjelmointikieli, jonka avulla verkkosivuista saadaan tehtyä interaktiiviset. JavaScript suoritetaan käyttäjän selaimessa, ei palvelimella, kuten esimerkiksi PHP-koodit. Tämä mahdollistaa nopeat ja välittömät muutokset verkkosivulla, koska sivua ei tarvitse ladata uudelleen koodin suorittamiseksi. (About.com 2016.)

HTML-sivussa JavaScript kirjoitetaan script-tagin sisään, tai se voidaan ladata ulkopuolisesta lähteestä script-tagin src-attribuutilla (source), esimerkiksi <script src="myjavascript.js" />. JavaScript-tiedostoissa käytetään päätettä ".js". JavaScript-koodi sijoitetaan joko sivuston head- tai body-tagiin. Jos skripti sijoitetaan body-tagin sisään, se kannattaa sijoittaa sen loppuun, koska JavaScript-koodin lataus voi hidastaa sivuston näkyvän osuuden latausta. Jos koodi sisältää DOM-muutoksia, eli HTML-elementtien manipulointia, elementtien tulisi olla ladattuna ennen JavaScript-koodin suoritusta.

Opinnäytetyöprojektissa puhdasta JavaScript-koodia ei käytetä juuri lainkaan. Koodi sisältää kyllä JavaScript-funktioita, mutta ne sisältävät pääasiassa jQuery-koodia. JavaScriptiä käytetään lähinnä siis jQueryn ja muiden JavaScript-kirjastojen kautta.

```
<button id="painike">Vaihda taustaväri punaiseksi!</button>

<script>

    document.getElementById("painike").addEventListener('click', function(){
        document.body.style.background = "red";
    });

</script>
```

Kuva 11. Esimerkki JavaScript-koodista, joka vaihtaa sivuston taustavärin punaiseksi painiketta painamalla.

4.1.6 MySQL

MySQL on Oracle Corporationin kehittämä, maailman suosituin avoimen lähdekoodin SQL-tietokantajärjestelmä (MySQL.com n.d.). Lyhenne SQL tulee sanoista Structured Query Language. Tietokantojen avulla suuriakin tietomääriä voidaan tallentaa jäsennellyksi kokoelmaksi, josta tietoa voidaan hakea ja muokata helposti kyselylausekkeilla.

MySQL-tietokantoja voidaan käyttää lähes millä ohjelmointikielellä tahansa. Opinnäytetyöprojektissa tietokantojen hallinnointiin käytetään PHP-kieltä. Projektissa tietokantoja käytetään käyttäjätietojen ja teemojen säilytykseen.

Tietokannat koostuvat tauluista ja niiden sisällä olevista sarakkeista ja riveistä. Tauluja voidaan yhdistää toisiinsa tunnisteiden avulla.

Taulukko 1. Esimerkki tietokantataulusta nimeltään themeslots.

themeslots			
ID	**SteamID**	**amount**	**updated**
1	76561198038927965	4	2013-04-07 10:16:24
2	76521798418927910	2	2013-09-24 18:00:10
3	76591497518627272	3	2014-03-16 15:45:59
4	76548531997927962	2	2015-07-28 22:46:00
5	76561898038924612	1	2016-09-02 08:30:00

Taulukossa 1 on uudessa web-sovelluksessa käytetty tietokantataulu, joka sisältää määrän, kuinka monta ylimääräistä teemaa käyttäjä voi sovelluksella tehdä. Ylimääräisiä teemapaikkoja annetaan lahjoittajille.

ID-sarakkeessa on yksilöivä arvo, jolle on asetettu lisämääre ”AUTO_INCREMENT”, jolloin ID nousee aina yhdellä ylöspäin, kun uusi tietokantarivi lisätään. SteamID-sarake sisältää käyttäjän Steam-tunnisteen. Amount-sarake sisältää ylimääräisten teemapaikkojen määrän. Updated-sarakkeessa on päivämäärä ja kellonaika, milloin tietokantariviä on viimeksi muokattu. Updated-sarakkeelle on annettu lisämääre ”on update CURRENT_TIMESTAMP”, jolloin kenttä päivittyy automaattisesti.

Tietokantarivejä lisätään INSERT INTO -kyselyn avulla. Seuraava kysely lisää rivin Steam-tunnisteella 76561198038927965 ja teemapaikkojen määräksi asetetaan 4.

```
INSERT INTO themeslots (steamID, amount)
VALUES (”76561198038927965”, 4);
```

Tietokantarivien hakeminen tietokannasta tapahtuu SELECT-kyselyiden avulla. Seuraava kysely hakee themeslots-taulusta kaikki rivit, joissa teemapaikkoja on tasan kaksi. Tähti tarkoittaa sitä, että kysely palauttaa kaikki rivin sarakkeet. Tähän voidaan listata pilkulla eroteltuna halutut sarakkeet, jos kaikkia ei haluta palauttaa.

```
SELECT * FROM themeslots WHERE amount = 2;
```

Tietokantarivien päivitys tapahtuu UPDATE-kyselyn avulla. Seuraava kysely päivittää teemapaikkojen määräksi kolme niissä riveissä, joissa Steam-tunniste on 76561198038927965.

```
UPDATE themeslots SET amount = 3
WHERE SteamID = ”76561198038927965”;
```

Tietokantarivejä voidaan poistaa DELETE-kyselyllä. Seuraava kysely poistaa kaikki rivit, joita on viimeksi päivitetty vuonna 2016.

```
DELETE FROM themeslots WHERE YEAR(updated) = 2016;
```

4.2 Käytetyt kirjastot

Web-sovelluksessa käytettiin useita JavaScript-kirjastoja. Tässä luvussa esitellään niistä tärkeimmät.

4.2.1 jQuery

jQuery on kevyt JavaScript-kirjasto, joka mahdollistaa JavaScript-komentojen kirjoittamisen vähemmällä määrällä koodia ja helpommin ymmärrettävässä muodossa. jQuery on helpompi oppia kuin JavaScript sen yksinkertaisuutensa vuoksi. Käyttäjällä tulisi kuitenkin olla JavaScriptin perusteet hallussa ennen jQueryn käyttöönottoa.

jQuery on suosittu kirjasto, koska se yksinkertaistaa mm. DOM-manipuloinnin, tapahtumankäsittelijöiden lisäyksen sekä AJAX-kyselyiden ja animaatiotehosteiden käytön (Tutorialspoint.com n.d.b). DOM-manipuloinnilla tarkoitetaan HTML-elementtien hallintaa. AJAX-kyselyiden avulla tietoa voidaan ladata ulkopuolisista lähteistä. Tapahtumankäsittelijöiden avulla elementteihin saadaan lisättyä interaktiivisia toiminnollisuuksia, kuten mitä esimerkiksi tapahtuu, kun elementtiä klikataan tai kun hiiren kursori viedään elementin päälle.

jQuery-kirjasto on JavaScript-tiedosto, joka tulee ladata sivulle joko ulkopuolisesta lähteestä tai omalta palvelimelta.

```
<button id="painike">Vaihda taustaväri punaiseksi!</button>

<script>

    $('#painike').click(function(){
        $('body').css('background', 'red');
    });

</script>
```

Kuva 12. Esimerkki jQuery-koodista, joka vaihtaa sivuston taustavärin punaiseksi painiketta painamalla.

4.2.2 JSZip

JSZip on JavaScript-kirjasto, joka mahdollistaa ZIP-pakettien luomisen, lukemisen ja muokkauksen helppokäyttöisen rajapinnan avulla (JSZip on Github 2016).

JSZip toimii eniten käytettyjen selainten kanssa. Safari-selain ei jostain syystä palauta ZIP-pakettia käyttäjälle, vaikka JSZipin sivusto väittää kirjaston toimivan myös kyseisellä selaimella.

JSZip oli moduuli, joka mahdollisti HUD-editorin toteutuksen. Ennen JSZip-kirjaston löytymistä, minulla ei ollut tiedossa tapaa, miten käyttäjä voisi muokata tiedostoja ja palauttaa muutokset itselleen niin, ettei se rasittaisi palvelinta. Koska JSZip on JavaScript-kirjasto, kaikki muokkaukset tapahtuvat käyttäjän selaimessa, eikä näin rasita toonhud.com-sivuston palvelinta.

JSZip-kirjasto päivittyi versioon 3 web-sovelluksen toteutuksen ja opinnäytetyön kirjoituksen välissä. Päivitys muutti JSZipin useita metodeja, tehden niistä asynkronisia. Asynkroninen tarkoittaa sitä, että koodi suoritetaan taustalla, niin ettei se keskeytä muita toimintoja. Kuvan 13 esimerkki on kirjoitettu päivitetyllä versiolla (3.1.3).

```
// Lataa olemassa oleva ZIP-paketti Paketti.zip
JSZipUtils.getBinaryContent('Paketti.zip', function(err, data) {

    // Lue ZIP-paketin sisältö
    JSZip.loadAsync(data)

    // Lisää teksti "Hello World!" tiedostoon Muokkaa.txt
    .then(function (zip) {

        var content = zip.file("Muokkaa.txt").async("text").then(function (txt) {
            return txt + "Hello World!";
        });
        zip.file("Muokkaa.txt", content);
        return zip;

    })

    // Poista tiedosto Poista.txt
    .then(function (zip) {

        zip.remove("Poista.txt");
        return zip;

    })

    // Muunna ZIP-paketti blob-muotoon
    .then(function (zip) {

        return zip.generateAsync({type: "blob"});

    })

    // Palauta päivitetty ZIP-paketti käyttäjälle nimellä Uusi_Paketti.zip
    .then(function (blob) {

        saveAs(blob, "Uusi_Paketti.zip");

    });

});
```

Kuva 13. Esimerkki ZIP-paketin muokkauksesta JSZip-kirjaston avulla.

Kuvan 13 koodi lukee ensin ZIP-tiedoston Paketti.zip. Sitten se lisää tiedoston Muokkaa.txt loppuun tekstin ”Hello World!”. Sen jälkeen ZIP-paketista poistetaan tiedosto Poista.txt. Lopuksi koodi palauttaa muokatun ZIP-paketin käyttäjälle nimellä Uusi_Paketti.zip.

4.2.3 Conditionizr

Koska JSZip-kirjasto ei vaikuta toimivan Safari-selaimella, tarvitsin tavan tunnistaa käyttäjät, jotka saapuvat web-sovellukseen kyseisellä selaimella. Selaintunnistukseen löytyi JavaScript-kirjasto nimeltään Conditionizr.

Conditionizr-kirjaston avulla voidaan tehdä erilaisia käyttäjäkohtaisia tarkistuksia, kuten tarkistaa mitä selainta tai käyttöjärjestelmää käyttäjä käyttää, onko käyttäjällä kosketusnäyttö tai mikä on hänen näyttönsä resoluutio. Conditionizr:iin voidaan lisätä helposti myös omia kyllä/ei-tarkistuksia.

Sivulle, jossa tarkistuksia tehdään, tulee ladata conditionizr.js-tiedosto. Conditionizr:n sisältämät tarkistukset ovat omissa tiedostoissaan, joten jos haluat tarkistaa käyttääkö käyttäjä esimerkiksi Safari-selainta, sivulle tulee ladata myös safari.js-tiedosto.

```
// Lisää Safari-selaimen tarkistus (löytyy safari.js-tiedostosta)
conditionizr.add('safari', function () {
    return /constructor/i.test(window.HTMLElement);
});


// Callback-funktio Safarin käyttäjille
conditionizr.on('safari', function () {
    // Käyttäjä käyttää Safaria - tee jotain
});


// Ehto if-lauseessa
if (conditionizr.safari)
{
    // Käyttäjä käyttää Safaria - tee jotain
}
else
{
    // Käyttäjä ei käytä Safaria - tee jotain
}
```

Kuva 14. Safari-selaimen tarkistus Conditionizr:n avulla.

5 SOVELLUKSEN TOTEUTUS

Tässä luvussa käydään tarkemmin läpi, kuinka web-sovelluksen tärkeimmät ominaisuudet toteutettiin.

5.1 Tunnistautuminen

Sovelluksessa tehdyt teemat ovat yksilöllisiä. Siksi käyttäjien on tunnistauduttava, eli kirjauduttava sisään sivustolle. Tunnistautuminen antaa mahdollisuuden myös hallita käyttäjiä, kuten palkita lahjoittajia sekä seurata kävijämääriä.

Tunnistautuessa käyttäjää pyydetään yleensä antamaan jotain luottamuksellista, kuten sähköposti ja salasana. Käyttäjät odottavat, että sivusto pitää annetut tiedot itsellään, eikä levitä tietoja kenellekään. Se on suuri vastuu. Tietoturvan tulee olla kunnossa. Tietomurtoja kuitenkin tapahtuu. Paras tapa on siirtää vastuu itseltä pois ja ulkoistaa kirjautuminen kolmannelle osapuolelle. Tässä apuun tulee OpenID.

5.1.1 OpenID

OpenID on avoimen lähdekoodin palvelu, joka mahdollistaa kirjautumisen sivustolle käyttäen toisen palvelun tunnusta. Näin käyttäjän ei tarvitse luoda aina uutta käyttäjätunnusta jokaiselle sivustolle, vaan hän voi käyttää tunnusta, jonka hän on jo luonut toiseen palveluun. Monet palvelut tarjoavat OpenID-tunnistautumista, kuten Facebook, Yahoo!, Microsoft ja Google. Näitä palveluita kutsutaan identiteetin tarjoajiksi (identity provider). Kun käyttäjä kirjautuu sivustolle OpenID:n avulla, kirjautuminen tapahtuu identiteetin tarjoajan sivustolla, jättäen käyttäjätunnuksen ja salasanan tarkistuksen heille. Identiteetin tarjoajat voivat itse määrittää mitä tietoa palautetaan takaisin sivustolle, josta alun perin kirjauduttiin sisään. (OpenID n.d.)

Myös Steam tarjoaa OpenID-tunnistautumista, antaen kehittäjille mahdollisuuden lisätä Steam-kirjautumisen omille web-sivustoilleen (Steam n.d.c). Tässä projektissa käytettiin Steamia kirjautumiseen. Steam-kirjautuminen oli luonteva valinta, koska sovelluksen käyttäjäryhmä, eli Team Fortress 2 -pelin pelaajat, ovat joka tapauksessa jo kirjautuneet Steamiin, koska pelin pelaaminen vaatii sitä.

5.1.2 Kirjautuminen

Steam-kirjautumiseen käytettiin pohjana AlliedModders.com-sivuston keskustelupalstalta löytynyttä esimerkkikoodia. Koodi on kirjoitettu PHP-

kielellä. Koodi sisältää LightOpenID-kirjaston, sekä esimerkin, kuinka kirjautuminen onnistuu kirjastoa käyttäen (Jankec90 2013). Koodia muokattiin sovelluksen tarpeisiin sopivaksi.

Kun toonhud.com-sivustolla painetaan kirjaudu sisään -painiketta, selain ohjautuu Steamin sivustolle, jossa kirjautuminen tapahtuu. Kun käyttäjä kirjautuu sisään omalla Steam-tunnuksellaan, hänet ohjataan takaisin toonhud.com-sivustolle. Steam palauttaa toonhud.com-sivustolle vain käyttäjän 64-bittisen SteamID:n, joka on käyttäjäkohtainen, mutta julkinen, 17 numeroinen numerosarja.

SteamID:stä luodaan käyttäjälle salattu lause. Ensin luodaan teksti, joka sisältää käyttäjän SteamID:n, IP-osoitteen sekä jonkun kirjain-numeroyhdistelmän. Lopuksi teksti salataan käyttäen MD5-kryptausta. Näin käyttäjälle saadaan salattu lause, joka on henkilökohtainen, eikä muiden käyttäjien pääteltävissä.

Salattu lause tallennetaan käyttäjän selaimeen evästeeseen. Eväste tallennetaan HttpOnly-attribuutin kanssa, jolloin evästettä voi muokata käytännössä vain toonhud.com-sivusto. Salattu lause tallennetaan myös sivuston tietokantaan. Tietokantatauluun toonhud_login tallennetaan käyttäjän SteamID, salattu lause sekä päivämäärä, jolloin rivi lisättiin. Kun käyttäjä avaa minkä tahansa sivun toonhud.com-sivustolla, sivu vertaa evästeessä olevaa salattua lausetta tietokannassa olevaan. Jos tietokannasta löytyy evästeessä oleva salattu lause ja tietokantaan tallennettu salattu lause on lisätty alle 2 viikkoa sitten, sivusto näyttää käyttäjän sisään kirjautuneena.

Steam-palvelussa käyttäjillä on oma nimimerkki sekä avatar-kuva. Näitä haluttiin käyttää myös toonhud.com-sivustolla. Kirjautumisen yhteydessä näitä tietoja Steam ei palauta, joten ne on haettava erikseen. Tässä apuun tulee Steamin rajapinta: Steam Web API.

5.1.3 Steam Web API

Steam tarjoaa kehittäjille web-rajapinnan, jonka avulla kehittäjät voivat hakea sivustoilleen tai sovelluksiinsa tietoa, kuten pelikohtaisia uutisia, pelien statistiikkaa sekä julkisia käyttäjätietoja Steamista. Kehittäjän tulee olla rekisteröitynyt Steamiin ja hänen tulee hyväksyä rajapinnan käyttöehdot sekä luoda henkilökohtainen API-avain Steamin sivustolla. API-avainta käytetään, jotta Steam voi yhdistää rajapintakutsut tiettyyn käyttäjään. Steam voi tarvittaessa mitätöidä API-avaimen, jos rajapintaa käyttävä sovellus ei noudata Steam web-rajapinnan käyttöehtoja. (Steam n.d.c.)

Web-sovelluksessa käytetään Steam web-rajapinnan metodia getPlayerSummaries, jonka avulla saadaan haettua käyttäjäkohtaisia tietoja. Sovellukseen haetaan käyttäjän nimimerkki sekä kaksi erikokoista avatar-kuvaa (kuvien URL-osoitteet). Ensimmäisen kerran tiedot haetaan kirjautumisen

yhteydessä. Tiedot tallennetaan tietokantatauluun toonhud_user_cache, joka sisältää Steamista haetut käyttäjäkohtaiset tiedot.

5.1.4 ToonHUD-käyttäjänimi

Kun käyttäjä kirjautuu ensimmäisen kerran toonhud.com-sivustolle, häntä pyydetään keksimään käyttäjänimi itselleen. Käyttäjänimeä käytetään vain käyttäjän ToonHUD-profiilin URL-osoitteessa (toonhud.com/user/käyttäjänimi). Tässä olisi voitu käyttää myös esimerkiksi käyttäjän SteamID:tä, mutta selkeä käyttäjänimi URL-osoitteessa on aina siistimpi ja helpompi kirjoittaa sekä tunnistaa. Käyttäjänimi tallennetaan toonhud_users tietokantatauluun. Tauluun tallennetaan myös käyttäjän rekisteröintipäivämäärä sekä erinäisiä käyttäjäkohtaisia asetuksia.

5.1.5 Käyttäjätietojen päivitys

Kun käyttäjä muuttaa nimimerkkiään tai avatar-kuvaansa Steamissa, päivitetyt tiedot tulisi saada myös toonhud.com-sivustolle. Tämä toteutettiin jQuery-kirjastoa käyttäen AJAX-kyselyllä.

Kun käyttäjä avaa oman tai toisen käyttäjän profiilin sovelluksessa, sivulatauksen yhteydessä tarkistetaan, milloin tarkasteltavan profiilin omistajan Steam-tiedot on viimeksi päivitetty. Jos tiedot ovat yli 6 tuntia vanhoja, tiedot päivitetään.

Profiilin omistajan Steam-tiedot päivitetään suorittamalla JavaScript-funktio getSteamInfo. Funktiolle annetaan parametrina profiilin omistajan SteamID. Ensiksi funktio lisää profiilin omistajan avatar-kuvan päälle indikaattorin, joka ilmaisee, että profiilitietoja päivitetään. Indikaattorina toimii pyörivä reunus avatar-kuvan ympärillä. Sen jälkeen kutsutaan AJAX-kyselyllä PHP-tiedostoa loadSteamInfo.php. PHP-tiedostolle välitetään POST-metodilla profiilin omistajan SteamID. PHP-tiedosto hakee Steamista web-rajapintaa käyttäen SteamID:n perusteella löytyvän käyttäjän tiedot, eli nimimerkin sekä kaksi erikokoista avatar-kuvaa. PHP-tiedosto päivittää tiedot tietokantaan ja palauttaa tiedot takaisin JavaScript-funktiolle. Onnistuneen AJAX-kyselyn jälkeen JavaScript-funktio vaihtaa sivulla näkyvän avatar-kuvan sekä käyttäjän nimimerkin päivitettyyn tietoon. Lopuksi funktio poistaa päivitystä ilmaisevan indikaattorin.

```
function getSteamInfo(id) {

    // Näytä latausindikaattori avatar-kuvan ympärillä
    $('#profilebox #avatar .spinner').show();

    // Kutsu loadSteamInfo.php -tiedostoa, joka hakee käyttäjän tiedot Steamistä,
    // tallentaa ne tietokantaan ja palauttaa tiedot AJAX-kyselyyn
    $.ajax({
        type: "POST",
        url: "/ajax/loadSteamInfo.php",
        data: {'steamID' : id}, // Lähetä käyttäjän SteamID PHP-tiedostolle POST-metodilla
        cache: false,
        dataType: 'json', // Palauta tiedot JSON-muodossa
        timeout: 6000,
        success: function(data) {

            // AJAX-kysely onnistui

            // Päivitä sivulla näkyvä avatar-kuva sekä käyttäjänimi
            $('#profilebox #avatar').css("background-image","url('"+data.avatarfull+"')");
            $('#profilebox #name').text(data.username);

            // Jos kyseessä oli omien tietojen päivitys,
            // päivitä nimimerkki myös sivuston yläpalkista
            if (data.me) { $('#userInfo #name a').text(data.username); }

        },
        complete: function(data) {

            // AJAX-kysely valmis

            // Piilota latausindikaattori
            $('#profilebox #avatar .spinner').hide();

        }
    });
}
```

Kuva 16. JavaScript-funktio getSteamInfo.

5.2 **Teemat**

Sovelluksella tehtyjä ToonHUD-muokkauksia kutsutaan teemoiksi. Käyttäjät pystyvät lataamaan toisten käyttäjien teemoja. Sovelluksesta löytyy myös valmiiksi tehtyjä teemoja, joita käyttäjät voivat ladata. Vain kirjautuneet käyttäjät pääsevät luomaan omia teemoja. Kirjautuneilla käyttäjillä on mahdollisuus luoda kaksi teemaa. Lahjoittajille tarjotaan useampia teemapaikkoja. Kirjautuneet käyttäjät voivat monistaa omia teemojaan sekä kopioida toisten käyttäjien teemoja itselleen, jonka jälkeen he voivat muokata niitä mieleisekseen. Teemat voidaan asettaa yksityisiksi, jos käyttäjä ei halua muiden näkevän teemansa.

Opinnäytetyön liitteistä löytyy vuokaaviot, jotka kuvaavat seuraavissa luvuissa kerrottuja toimintoja. Niitä voi käyttää lukemisen tukena.

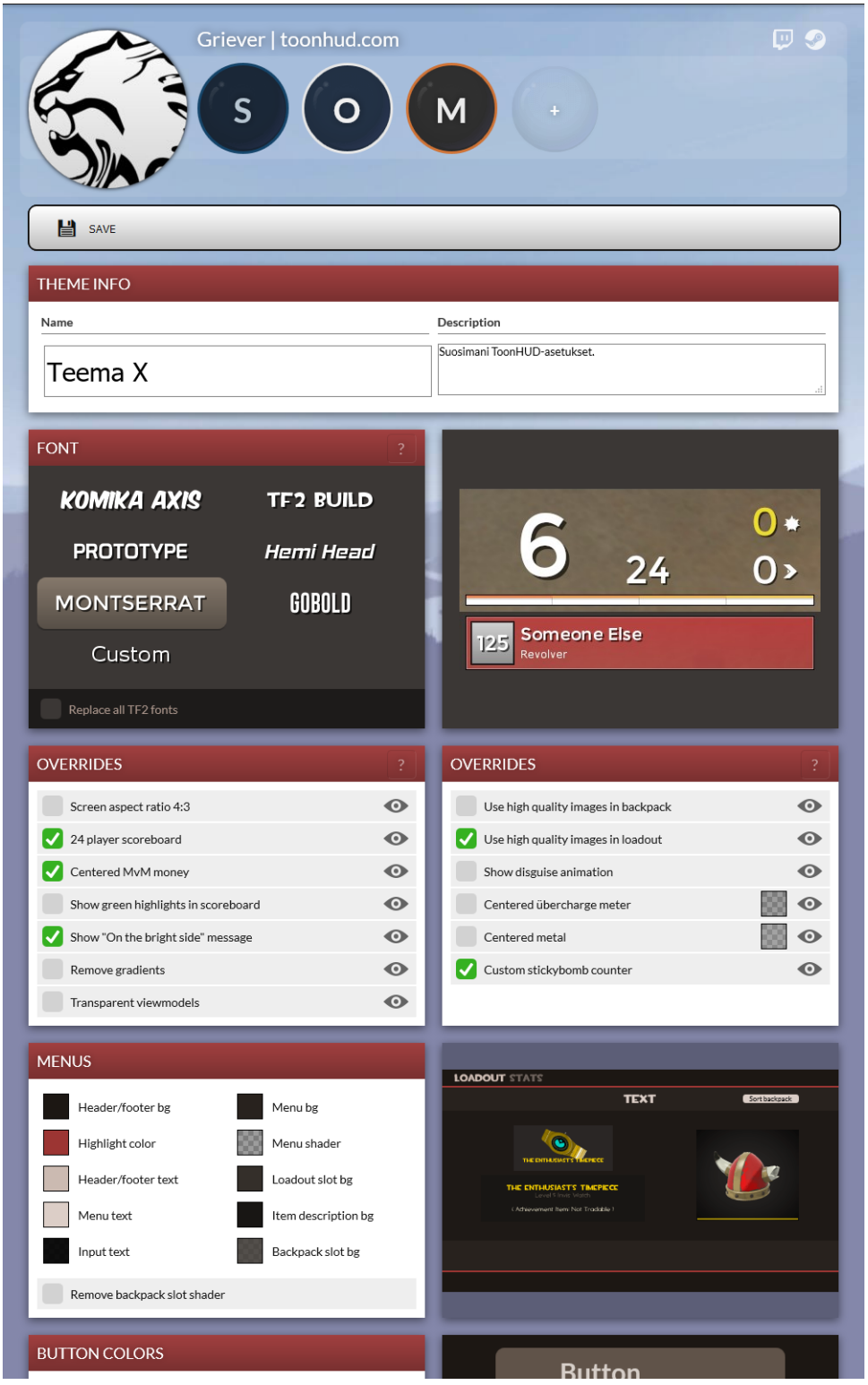
5.2.1 Uuden teeman luonti

Uuden teeman luonti alkaa käyttäjän profiilisivulta. Profiilisivulta löytyy käyttäjän nimi, avatar-kuva sekä lista käyttäjän tekemistä teemoista. Teemalistasta löytyy plus-painike, jos käyttäjällä on vapaita teemapaikkoja jäljellä. Painiketta painamalla päästään uuden teeman luontiin. Painike avaa URL-osoitteen toonhud.com/user/käyttäjänimi/theme/new. URL-osoite sisältää parametrin ”theme”, jonka arvo on ”new”. Tästä tunnistetaan,

että teema on uusi. Olemassa olevaa teemaa muokatessa theme-parametrin arvo on teeman ID. Kun URL-osoite sisältää theme-parametrin, sivu suorittaa JavaScript-funktion setTheme. Theme-parametri lähetetään setTheme-funktiolle parametrina.

setTheme-funktio lataa ensin AJAX-kyselyllä itse editorin, eli HTML-lomakkeen, joka sisältää kaikki valinnat ja esikatseluikkunat. Lomake ladataan tiedostosta editorForm.php. Kun lomake on ladattu, tarkistetaan, onko avattava teema uusi vai jo olemassa oleva. Tässä tapauksessa teema on uusi. Seuraavaksi lomakkeeseen lisätään kaikki jQuery-tapahtumankäsittelijät, joita lomakkeessa käytetään. Tapahtumankäsittelijöitä käytetään pääasiassa muutosten esikatseluikkunoissa. Näin muutokset saadaan heti näkyviin esikatseluun, kun käyttäjä tekee muutoksia editorissa.

Kun tapahtumankäsittelijät on lisätty lomakkeeseen, lomake lisätään HTML-dokumenttiin piilotettuna, jonka jälkeen se tuodaan näkyviin käyttäjälle liukutehosteen kera. Nyt käyttäjä voi tehdä haluamansa muutokset Toonhudiin.



Kuva 17. Web-sovelluksen editorinäkymä.

5.2.2 Teeman tallennus

Käyttäjä voi tallentaa editorilla tekemänsä muutokset Save-painiketta painamalla. Painiketta painamalla se kytkeytyy pois käytöstä, jottei sitä voida

painaa kesken tallennuksen suorituksen. Sen jälkeen käynnistyy tarkistus, joka tutkii, onko kaikki pakolliset kentät täytetty. Tällä hetkellä ainoa pakollinen kenttä lomakkeessa on teeman nimi. Jos teeman nimeä ei ole syötetty, käyttäjälle näytetään virheilmoitus ja tallennus keskeytetään. Jos teemalle on annettu nimi, kutsutaan JavaScript-funktiota saveTheme.

saveTheme-funktio kutsuu AJAX-kyselyllä tiedostoa saveTheme.php. PHP-tiedostolle lähetetään kaikki lomakkeen tiedot. Ensin PHP-tiedostossa tarkistetaan, että käyttäjä on varmasti kirjautunut sisään. Sen jälkeen tarkistetaan, ollaanko tallentamassa uutta teemaa vai muokkaamassa olemassa olevaa. Jos teema on uusi, tarkistetaan, onko käyttäjällä vapaita teemapaikkoja jäljellä. Jos ollaan tallentamassa muutoksia jo olemassa olevaan teemaan, tarkistetaan että käyttäjä varmasti omistaa kyseisen teeman. Jos tiedot eivät läpäise tarkistusta, käyttäjälle näytetään virheilmoitus ja tallennus keskeytetään.

Teemaan halutaan tallentaa vain muokatut tiedot. Eli ne, mitkä poikkeavat oletusarvoista. Jos kaikki arvot tallennettaisiin, se kuluttaisi turhaan resursseja ja tietokantatilaa. Jotta muokatut arvot voidaan erottaa oletusarvoista, meidän tulee tietää mitkä oletusarvot ovat. Tämän vuoksi oletusarvot on tallennettu omaan XML-tiedostoon nimeltään defaultTheme.xml. Kun lomakkeeseen lisätään uusia valintoja, tulee sen oletusarvo lisätä tähän XML-tiedostoon. Tiedostossa saveTheme.php luetaan defaultTheme.xml-tiedoston sisältö muuttujaan, käyttäen simplexml_load_file-funktiota. Se on osa PHP:n SimpleXML-lisäosaa, joka on oletuksena asennettuna PHP:hen uusimmissa versioissa. SimpleXML-lisäosan avulla XML-tiedostojen hallinta onnistuu PHP:lla helposti.

```
<?xml version="1.0" encoding="UTF-8"?>
<skin>
    <field id="font" type="radio">
        <value>Default</value>
    </field>
    <field id="customFontPrimarySizeTiny" type="number">
        <value>12</value>
    </field>
    <field id="colorKillfeedBlue" type="text">
        <value>83 155 242 255</value>
    </field>
    <field id="killfeedXpos" type="text">
        <value>r640</value>
    </field>
    <field id="killfeedFontAntialias" type="checkbox">
        <value>on</value>
    </field>
    <field id="damageIndicatorEnableCustomFont" type="checkbox">
        <value>off</value>
    </field>
    <field id="crosshair1Opacity" type="number">
        <value>255</value>
    </field>
</skin>
```

Kuva 18. Supistettu versio defaultTheme.xml tiedostosta.

Kun oletusarvot on luettu defaultTheme.xml-tiedostosta, luodaan pohja XML-dokumentille, johon lisätään oletusarvoista poikkeavat arvot. Pohja luodaan tekemällä uusi SimpleXMLElement-objekti muuttujaan. Poikkeavat arvot poimitaan käymällä kaikki lomakkeen tiedot yksitellen läpi foreach-silmukassa. Arvoa verrataan sen oletusarvoon. Jos arvot poikkeavat toisistaan, arvo lisätään luotuun SimpleXMLElement-objektiin. Kun silmukka on suoritettu loppuun, muuttujaan jää XML-dokumentti, joka sisältää kaikki poikkeavat arvot.

Kun tallennettava teema on uusi, teemalle luodaan yksilöivä tunniste. Tunnisteen luomiseen käytetään pohjana StackOveflow.com-sivustolta löytynyttä PHP-funktiota (gord 2009). Funktio käyttää pohjanaan PHP:n uniqid-funktiota, joka luo uniikin tekstin sen hetkisen kellonajan mukaan (PHP.net n.d.). Tunnisteeksi luodaan kahdeksan merkin pituinen kirjain-numeroyhdistelmä. Tunnisteena oltaisiin voitu käyttää myös tietokannan uniikkia rivitunnistetta, mutta tunnisteen rakenteesta haluttiin helpommin tunnistettava.

Lopuksi saveTheme.php-tiedosto tallentaa teeman tiedot tietokantaan. Seuraavassa taulukossa esitetään tietokantaan tallennettavat tiedot. Tiedot tallennetaan tietokantatauluun nimeltään toonhud_themes.

Taulukko 2. Esimerkki tietokantaan tallennettavista teeman arvoista.

Nimi	Arvo	Kuvaus
themeID	RD3JQMMP	Generoitu uniikki ID
steamID	76561198038927965	Teeman omistajan SteamID
xml	<?xml version="1.0" encoding="UTF-8"?> <skin> <field id="font" type="radio"> <value>Montserrat</value> </field> </skin>	XML-dokumentti teemaan muokatuista arvoista
name	Teema X	Teeman nimi
description	Esimerkkikuvaus.	Teeman kuvaus
color	156,53,51|40,34,30|220,206,199	Kolme väriarvoa teeman valikoista. Värejä käytetään teemalistauksessa.
private	0	Teeman näkyvyysarvo. 0 = julkinen 1 = yksityinen

		2 = valmisteema
position	0	Teeman sijainti teemalistauksessa. Käyttäjät voivat vaihtaa teemojen järjestystä.
created	2016-11-20 20:53:29	Teeman luontiaika.
edited	2016-11-20 20:53:29	Aika milloin teemaa on viimeksi muokattu.

Kun tiedot on tallennettu tietokantaan, saveTheme.php-tiedosto palauttaa teeman tunnisteen AJAX-kyselylle. Jos tallennettava teema on uusi, käyttäjä uudelleenohjataan teeman uuteen URL-osoitteeseen. Jos tallennettava teema on jo ennestään olemassa oleva, käyttäjän teemalista päivitetään AJAX-kyselyllä ja käyttäjälle annetaan ilmoitus onnistuneesta tallennuksesta. Lopuksi tallennuspainike aktivoidaan takaisin käyttöön.

5.2.3 Tallennetun teeman muokkaus

Käyttäjä pääsee muokkaamaan tekemäänsä teemaa valitsemalla sen teemalistastaan. Sivu ohjautuu osoitteeseen toonhud.com/user/käyttäjänimi/theme/RD3JQMMP, jossa loppuosa on teeman ID. Editorilomake ladataan samalla tapaa kuin uutta teemaa luodessa, mutta ennen kuin editori näytetään käyttäjälle, siihen lisätään teemaan tehdyt muutokset.

Jotta tiedämme, mitä muutoksia teema sisältää, teeman tiedot tulee ensin ladata tietokannasta. Tämä tapahtuu kutsumalla AJAX-kyselyn avulla tiedostoa getTheme.php. Tämä PHP-tiedosto hakee tietokannasta teeman annetun ID:n perusteella ja palauttaa sen XML-muodossa.

Muutokset lomakkeeseen tehdään laukaisemalla JavaScript-funktio setThemeForm, jolle annetaan parametreiksi teeman id, teeman XML-dokumentti sekä itse lomake. Funktio käy yksitellen läpi teeman XML-dokumentissa olevat arvot ja lisää ne lomakkeeseen ID:n mukaiseen kenttään. Esikatseluikkunoissa käytetyt arvot asetetaan myös tässä funktiossa. Lomaketta muokataan jQuery-kirjaston avulla.

5.2.4 Teeman kopiointi ja monistus

Kirjautuneet käyttäjät voivat kopioida toisten tekemiä teemoja itselleen sekä monistaa omia teemojaan. Kopiointi tapahtuu teeman editorinäkymässä Copy-painiketta painamalla. Jos ollaan omassa teemassa, painikkeen nimi on Duplicate. Käyttäjät näkevät painikkeen vain silloin, kun heillä on vapaita teemapaikkoja jäljellä. Painiketta painamalla se kytkeytyy pois

päältä, jottei sitä voida painaa kopioinnin ollessa käynnissä. Sen jälkeen painike suorittaa JavaScript-funktion copyTheme.

JavaScript-funktio copyTheme kutsuu AJAX-kyselyllä tiedostoa copyTheme.php. PHP-tiedostolle lähetetään kopioitavan teeman ID sekä käyttäjän Steam-tunniste. PHP-tiedosto copyTheme.php tarkistaa ensin, onko käyttäjällä varmasti vapaita teemapaikkoja jäljellä. Vaikka kopiointipainike on piilotettu editorissa, se ei estä kokenutta käyttäjää suorittamasta JavaScript-komentoja selaimella. Koska JavaScript-tarkistukset suoritetaan käyttäjän selaimessa, käyttäjät voivat manipuloida niitä. Siksi on tärkeää tehdä tarkistukset myös palvelimen puolella, eli PHP-tiedostossa.

Jos käyttäjällä on vapaita teemapaikkoja jäljellä, etsitään kopioitava teema tietokannasta sen ID:n perusteella. Jos teema löytyi, generoidaan teeman kopiolle oma ID. Sen jälkeen tietokantaan luodaan uusi tietokantarivi käyttämällä kopioitavan teeman tietoja. Tiedot, jotka vaihdetaan, on teeman ID, teeman omistajan Steam-tunniste sekä teeman luontiaika. Teema asetetaan myös yksityiseksi.

5.2.5 Teeman lataus

Sekä kirjautuneet että kirjautumattomat käyttäjät voivat ladata luotuja teemoja. Teeman lataus alkaa Download-painiketta painamalla. Ensiksi tarkistetaan, käyttääkö käyttäjä Safari-selainta. Tarkistus tehdään Conditionizr-kirjaston avulla. Jos Safari-selain on käytössä, avataan latausnäkymä ja ilmoitetaan että lataus ei onnistu kyseisellä selaimella. Muussa tapauksessa tarkistetaan, onko käyttäjä jo aloittanut latauksen, mutta sulkenut latausnäkymän. Tällaisessa tapauksessa Download-painike tuo latausnäkymän takaisin esiin. Jos aikaisempaa latausta ei ole käynnissä, merkitään lataus käynnistetyksi. Tämä tieto säilytetään JavaScript-muuttujassa. Ennen kuin latausnäkymä tuodaan esiin, nollataan siinä oleva konsoli. Konsoli on ruutu, johon kirjataan lataustapahtuman kulku. Siihen kirjataan kaikki muutokset mitä teeman asennus toonhud.zip-tiedostolle tekee.

Kuva 19. Web-sovelluksen latausnäkymä.

Koska käyttäjä voi suorittaa latauksia useita kertoja, on turhaa ladata samaa muokkaamatonta toonhud.zip-tiedostoa uudelleen ja uudelleen. Tästä syystä ensimmäisellä latauskerralla tiedosto tallennetaan käyttäjän selaimen omaan tietokantaan nimeltään IndexedDB. IndexedDB on HTML5:n mukana tullut ominaisuus, joka mahdollistaa suurienkin tietomäärien tallentamisen suoraan selaimeen ja tiedon käyttämisen saman domainin alla (Tutorialspoint.com n.d.a). IndexedDB toimii uusimmilla ja käytetyimmillä selaimilla. IndexedDB-tietokantaan tallennetaan myös Toonhudin versionumero, jotta sovellus osaa ladata uuden version sen ilmestyessä.

Kun latausnäkymä on avattu, suoritetaan JavaScript-funktio getZip. Ensiksi funktio tarkastaa, löytyykö toonhud.zip-tiedosto jo käyttäjän selaimen IndexedDB-tietokannasta ja vastaako versionumero tietokannassa olevaa. Jos tiedosto löytyy ja versionumerot vastaavat toisiaan, latausvaihe ohitetaan ja siirrytään suoraan teeman asennukseen. Jos tiedostoa ei löydy, siirrytään toonhud.zip-tiedoston lataukseen.

Lataus aloitetaan asettamalla latausnäkymän latausprosentti nollaan ja vaihtamalla latauksen tilaksi ”Downloading”. Sen jälkeen luetaan URL-osoite, josta toonhud.zip tullaan lataamaan. Osoite luetaan AJAX-kyselyllä tiedostosta dlsource.php. PHP-tiedosto tarkistaa, onko käyttäjä kirjautunut sisälle. Jos on, katsotaan käyttäjän asetuksista, minkä latauslähteen hän on valinnut käyttäjäasetuksissa. Käyttäjät voivat valita latauksen kahdesta eri lähteestä: Mediafire.com tai toonhud.com. Oletuksena käytetään Mediafiren palvelinta, koska tiedoston lataus haluttiin ulkoistaa, jottei omalle palvelimelle tule ylimääräistä kaistankäyttöä. Joissain maissa jotkut internetpalveluntarjoajat ovat estäneet Mediafire-palvelun käytön, minkä takia lisäsin vaihtoehtoiseksi latauslähteeksi sovelluksen oman palvelimen, toonhud.comin.

Kun latausosoite on selvillä, suoritetaan lataus käyttäen JSZip-kirjastoa. Latauksen aikana latausnäkymässä näkyvää latausprosenttia päivitetään. Jos latauksen aikana tapahtuu virhe, siitä ilmoitetaan käyttäjälle latausnäkymässä olevassa konsolissa. Jos lataus onnistui, selaimen IndexedDB-tietokanta tyhjennetään ja sinne tallennetaan uusi versionumero sekä ladattu toonhud.zip-tiedosto. Tämän jälkeen siirrytään asentamaan teemaa, eli lisäämään käyttäjän tekemiä muutoksia ladattuun zip-pakettiin. Latauksen tilaksi vaihdetaan ”Installing theme” ja suoritetaan JavaScript-funktio saveOverrides.

Funktion saveOverrides tarkoitus on muokata ladatun ZIP-paketin sisällä olevia tiedostoja lisäten teeman muutokset. Ensiksi funktio lukee overrides.xml-tiedoston sisällön AJAX-kyselyllä. Tämä XML-tiedosto kertoo mitä tiedostoja ZIP-paketista muokataan ja millä tavalla.

```
<?xml version="1.0" encoding="UTF-8"?>
<overrides>

    <override id="24PlayerScoreboard">
        <action type="replace">
            <new>toonhud/overrides/24 player scoreboard/resource/ui/scoreboard.res</new>
            <old>toonhud/resource/ui/scoreboard.res</old>
        </action>
    </override>

    <override id="HQloadoutImages">
        <action type="edit">
            <file>toonhud/resource/ui/classloadoutpanel.res</file>
            <comment>HQ images</comment>
            <value>1</value>
        </action>
        <action type="edit">
            <file>toonhud/resource/ui/itemselectionpanel.res</file>
            <comment>HQ images</comment>
            <value>1</value>
        </action>
    </override>

    <override id="colorMainMenuButton">
        <action type="color">
            <comment>Main Menu button color</comment>
        </action>
    </override>

</overrides>
```

Kuva 20. Supistettu versio tiedostosta overrides.xml.

Tiedosto overrides.xml koostuu override-tageista, jotka sisältävät attribuutin id, joka vastaa editorissa olevan kentän ID:tä. Override-tagi sisältää action-tageja, jotka kuvaavat yksittäistä muutosta. Action-tagit sisältävät attribuutin type, joka kertoo millä tavalla muutos tehdään tiedostoon. Action-tagi sisältää erilaisia tageja type-parametrista riippuen. Erilaiset type-parametrin arvot on kuvattu seuraavassa taulukossa.

Taulukko 3. Tiedoston overrides.xml action-tyypit.

Action type	Kuvaus
animationlength	Vaihtaa animaation keston pituutta
color	Vaihtaa väriarvoa
edit	Vaihtaa arvon riveiltä, joilta löytyy annettu kommentti.
remove	Poistaa annetun tiedoston
removefiles	Poistaa kaikki editorissa listatut tiedostot
removeline	Poistaa rivit, joilta löytyy annettu kommentti
replace	Korvaa tiedoston toisella
replaceword	Korvaa tietyt sanat toisella sanalla riveiltä, joilta löytyy annettu kommentti.

Kun overrides.xml tiedosto on luettu, haetaan ladattavan teeman tiedot. Ladattava teema haetaan AJAX-kyselyllä tiedostosta getTheme.php. Kysely palauttaa teeman tiedot XML-muodossa. Seuraavaksi käydään teeman XML:ssä olevat muutokset yksitellen läpi. Muutokselle etsitään samalla

id:llä oleva override-tagi overrides.xml-tiedostosta. Sen jälkeen override-tagin sisällä olevat action-tagit käydään yksitellen läpi, muokaten ZIP-pakettia action-tagin tyypin ja sen sisällä olevien arvojen mukaan. Kun yksittäinen muutos on tehty, se kirjataan ylös käyttäjän nähtäväksi latausnäkymän konsoliin.

Kun muutokset on tehty ZIP-pakettiin, se palautetaan käyttäjälle. Latauksen tilaksi asetetaan ”Ready” ja lataus merkitään päättyneeksi. Nyt käyttäjä voi purkaa ZIP-paketin, siirtää sen sisällä olevan toonhud-kansion Team Fortress 2 -pelin custom-kansioon ja aloittaa pelaamisen.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli toteuttaa HUD-editori web-sovelluksena. Tähän tavoitteeseen myös päästiin. Sovellus julkaistiin helmikuussa 2016. Yhdeksän kuukauden jälkeen sovelluksella oli yli 75 000 rekisteröitynyttä käyttäjää ja yli 82 000 luotua teemaa. Olen tyytyväinen uuteen sovellukseen.

Web-sovelluksen toteutus oli pitkä, mutta opettavainen projekti. Sovelluksen toteutukseen meni töiden ohessa noin vuosi. Toteutus sisälsi myös asioita, joita ei tässä opinnäytetyössä mainittu. Näihin kuului muun muassa käyttöliittymän suunnittelu ja toteutus, hakukoneen toteutus teemojen ja käyttäjien etsintään, sekä ylläpitotoimintoja, kuten versiohistorian hallinta ja teemapaikkojen lisäys lahjoittajille.

Projektia aloittaessa minulla oli aiempaa kokemusta web-ohjelmoinnista. HTML, CSS, PHP ja MySQL oli minulle ennestään tuttuja tekniikoita. JavaScriptistä ja jQuerystä minulla oli perusteet hallussa, mutta opinnäytetyön aikana niistä kertyi paljon lisää kokemusta. Lisäksi tutustuin moniin uusiin JavaScript-kirjastoihin projektin aikana. Uusina asioina minulle tuli myös kirjautumisen lisäys sivustolle kolmannen osapuolen palvelun kautta OpenID:n avulla, sekä Steam Web-rajapinnan käyttö PHP:lla.

Windows-sovellukseen verrattuna web-sovelluksen ylläpito tuntuu helpommalta. Kaikki muutokset saa kerralla kaikille käyttäjille näkyviin, kun Windows-sovelluksessa päivitys meni aina usean vaiheen kautta. Sovellus piti ensin päivittää, pakata ZIP-paketiksi, lähettää palvelimelle ja vaihtaa versionumero sivustolta. Tämän jälkeen käyttäjät näkivät muutokset vasta kun he olivat päivittäneet sovelluksensa.

Sain pääasiassa hyvää palautetta web-sovelluksesta Toonhudin käyttäjiltä. Pidetyimpiä ominaisuuksia oli sovelluksen helppo käyttöönotto, valintojen esikatselut sekä teemojen jako muiden pelaajien kesken. Joitain käyttäjiä harmitti, että hudin päivitys vaatii nyt enemmän manuaalista työtä kuin aiemmin.

LÄHTEET

About.com (2016). What is JavaScript? Viitattu 28.10.2016.
http://javascript.about.com/od/reference/p/javascript.htm

Margaret Rouse (2013). What is heads-up display? Viitattu 9.10.2016.
http://whatis.techtarget.com/definition/heads-up-display-HUD

Mark Masters (2013). Designing a HUD That Works for Your Game. Viitattu 17.11.2016.
http://blog.digitaltutors.com/designing-a-hud-that-works-for-your-game/

Melissa Loomis (2015). HUD Design: A Good HUD Is Hard To Find. Viitattu 17.11.2016.
https://gamerant.com/good-hud-design-229/

MySQL.com (n.d.). What is MySQL? Viitattu 28.10.2016.
https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

Nimimerkki gord (2009). Short unique id in php. Viitattu 20.11.2016.
http://stackoverflow.com/a/1516430

Nimimerkki Jankec90 (2013). Simple php steam login in your site. Viitattu 19.11.2016.
https://forums.alliedmods.net/showthread.php?t=206689

Official TF2 Wiki (2016a). Team Fortress 2. Viitattu 23.10.2016.
https://wiki.teamfortress.com/wiki/Team_Fortress_2

Official TF2 Wiki (2016b). List of game modes. Viitattu 23.10.2016.
https://wiki.teamfortress.com/wiki/List_of_game_modes

OpenID (n.d.). What is OpenID? Viitattu 13.11.2016.
http://openid.net/get-an-openid/what-is-openid/

Philip Roestamadji (2015). Website vs. Web Application – What's the Difference? Viitattu 18.11.2016.
https://lionandpanda.com/website-vs-web-application-whats-the-difference/

PHP.net (n.d.a). PHP: What is PHP? Viitattu 25.10.2016.
http://php.net/manual/en/intro-whatis.php

PHP.net (n.d.b). PHP: uniqid. Viitattu 20.11.2016.
http://php.net/uniqid

Steam (2016a). Player and Game Statistics. Viitattu 23.10.2016.

http://store.steampowered.com/stats/

Steam (n.d.b). The Steam Workshop for Team Fortress 2. Viitattu 23.10.2016.
https://steamcommunity.com/workshop/about/?appid=440

Steam (n.d.c). Steam Web API Documentation. Viitattu 13.11.2016.
https://steamcommunity.com/dev

Stuart Knightley (2016). JSZip on Github. Viitattu 28.10.2016.
https://stuk.github.io/jszip/

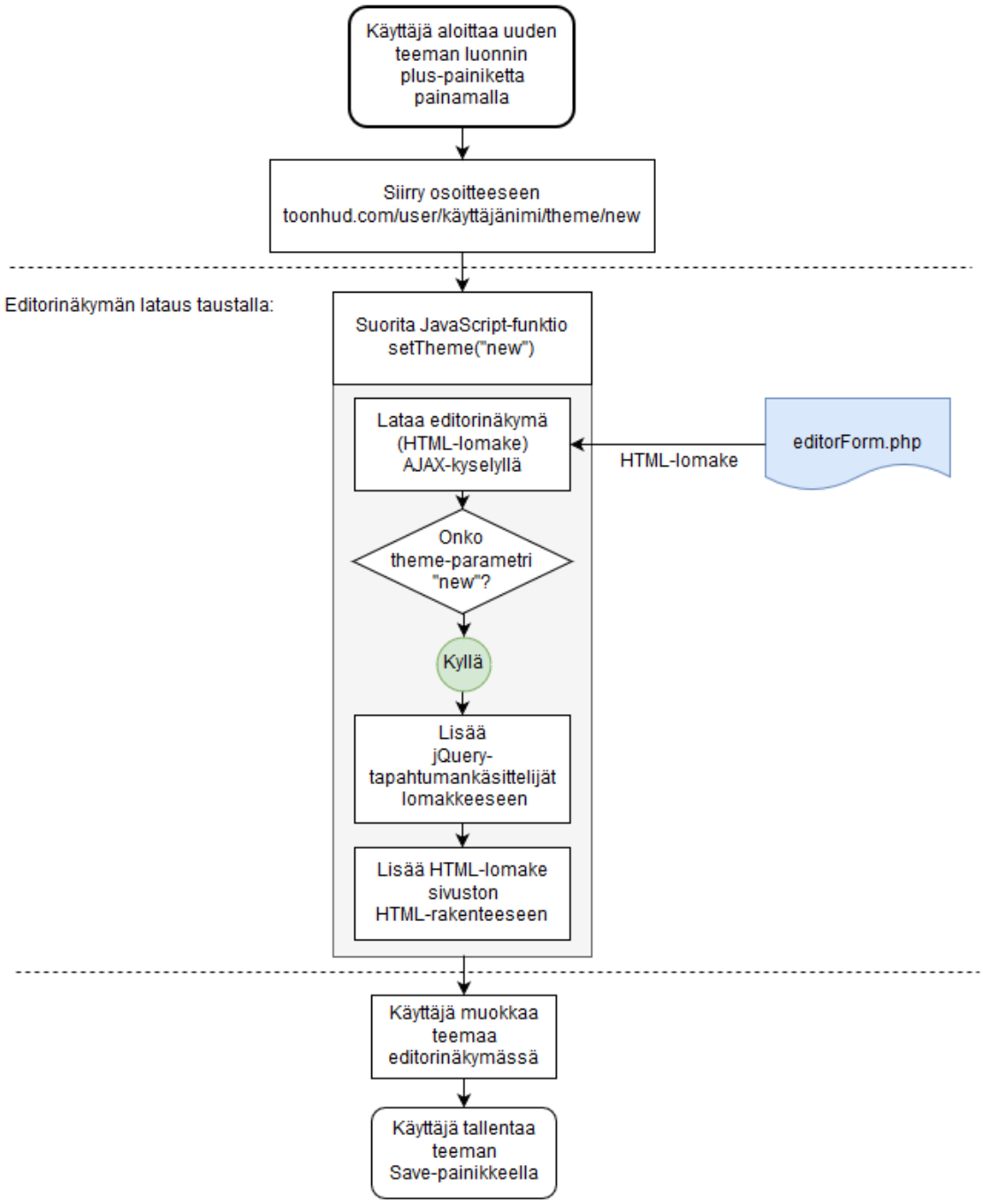
Tutorialspoint.com (n.d.a). HTML 5 – IndexedDB. Viitattu 2.12.2016
https://www.tutorialspoint.com/html5/html5_indexeddb.htm

Tutorialspoint.com (n.d.b). jQuery Overview. Viitattu 29.11.2016
https://www.tutorialspoint.com/jquery/jquery-overview.htm

W3Schools.net (n.d.). Introduction to XML. Viitattu 27.10.2016.
http://www.w3schools.com/xml/xml_whatis.asp

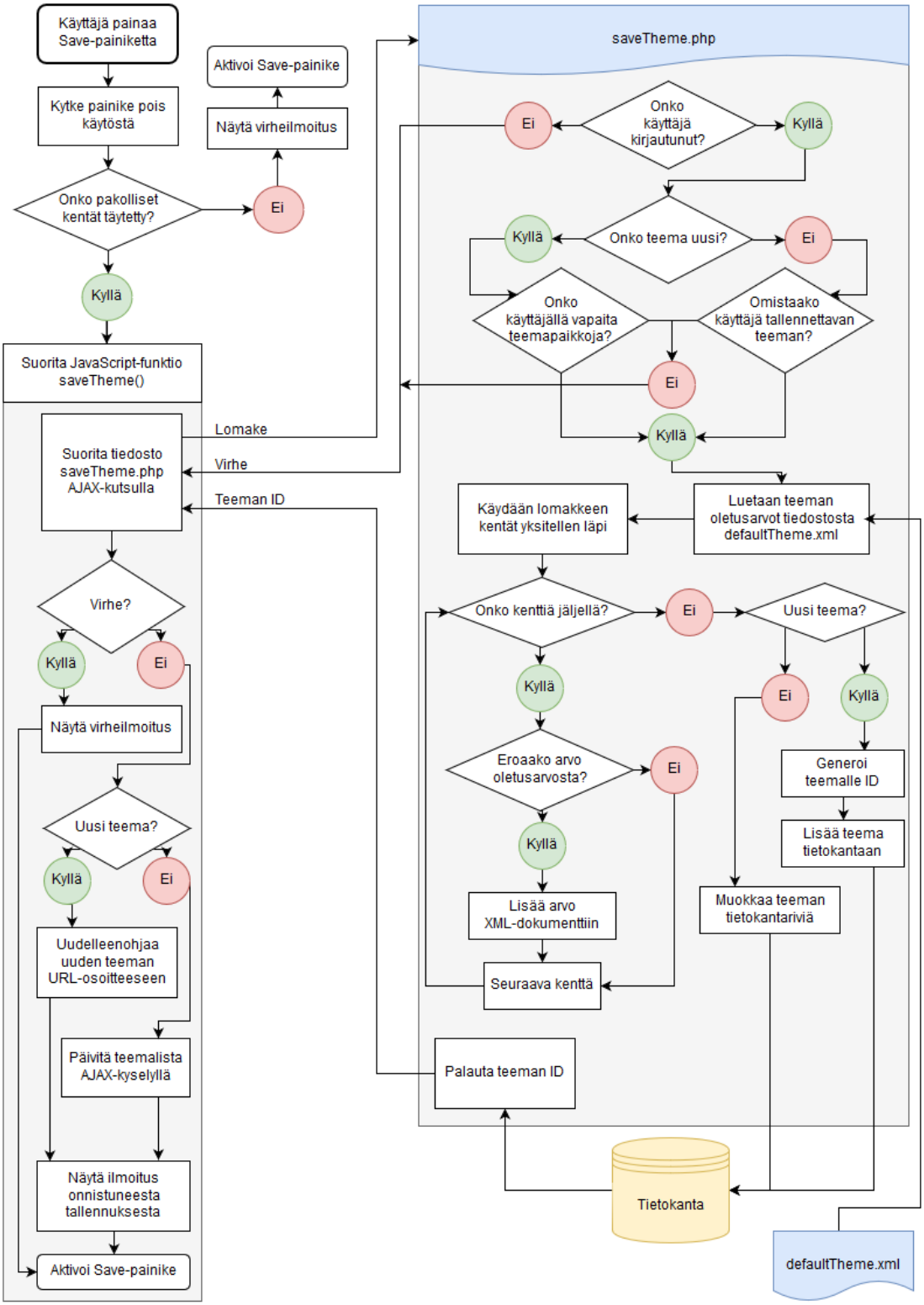
Liite 1

VUOKAAVIO UUDEN TEEMAN LUOMISESTA



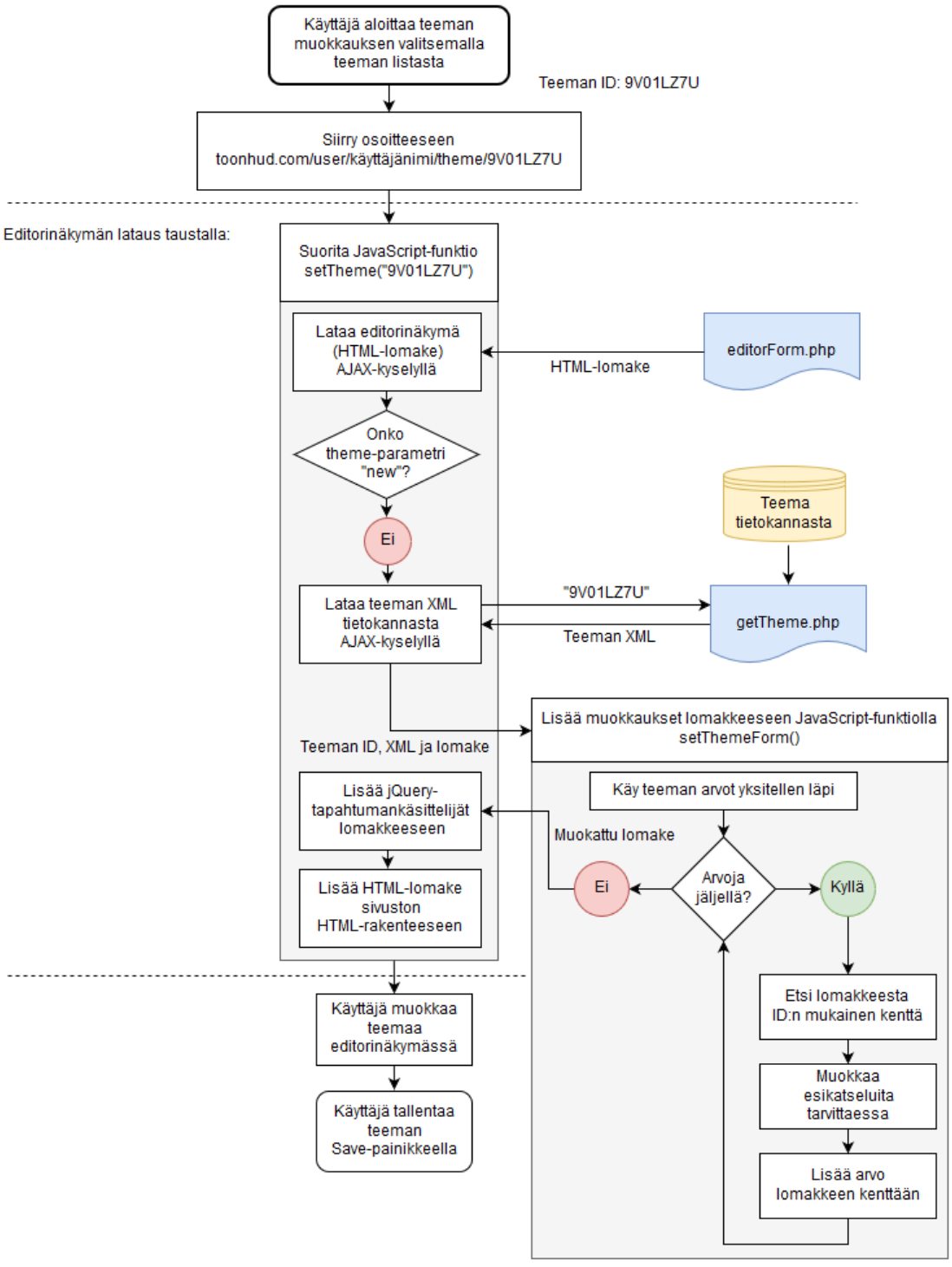
Liite 2

VUOKAAVIO TEEMAN TALLENNUKSESTA



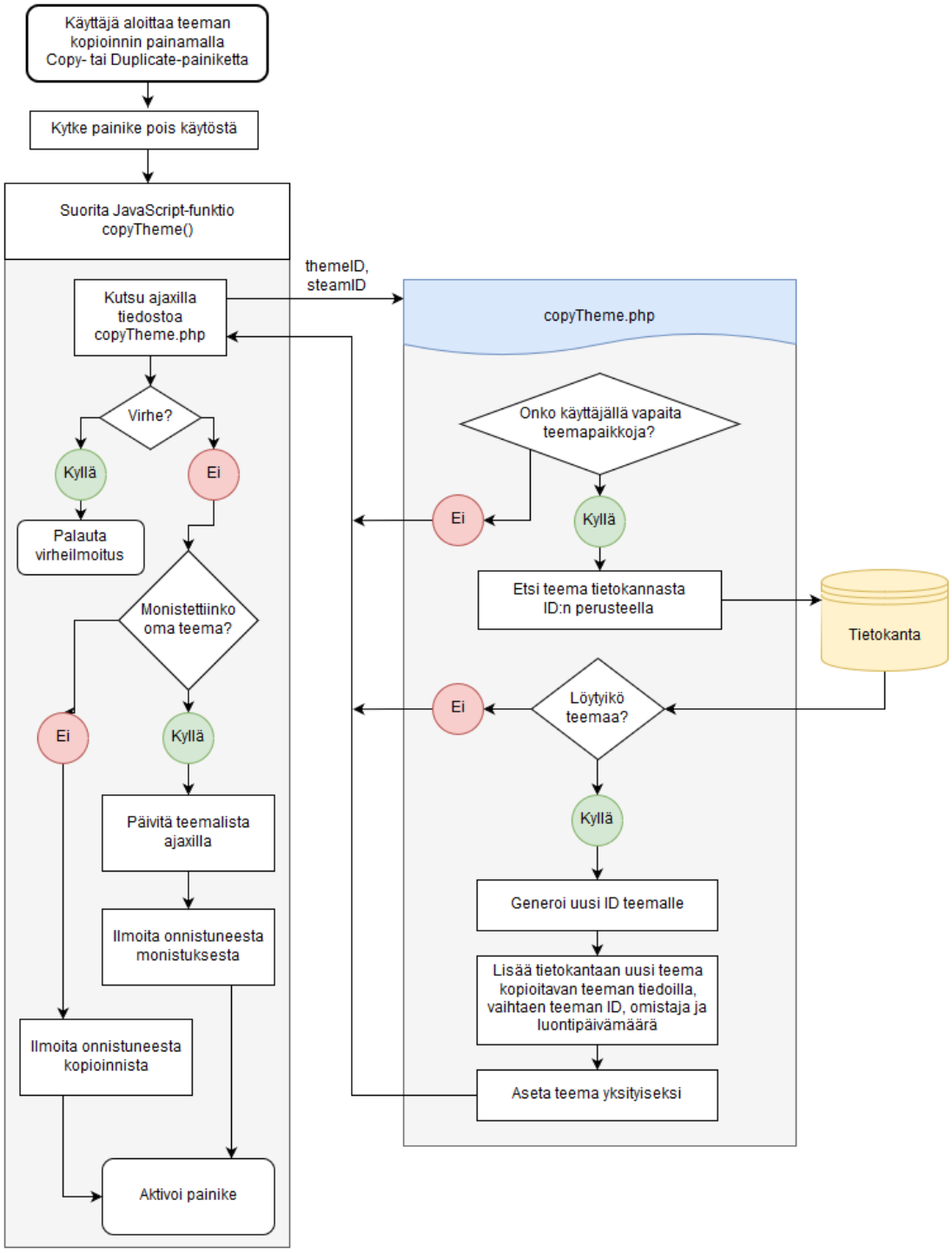
Liite 3

VUOKAAVIO TEEMAN MUOKKAUKSESTA



Liite 4

VUOKAAVIO TEEMAN KOPIOINNISTA/MONISTUKSESTA



Liite 5

VUOKAAVIO TEEMAN LATAUKSESTA