TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Tarkvarateaduse instituut

IT40LT

Janno Esko 134221IAPB

VEEBIRAKENDUSE ARENDAMINE *QUAKE 3* MOOTORIL PÕHINEVATE MÄNGUSERVERITE MAJUTAMISEKS *LINUX* SERVERITEL

Bakalaureusetöö

Juhendaja: Gert Kanter, MSc

Tallinn

2017

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Janno Esko

21.05.2017

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks oli luua rakendus, mis käsitleb *Quake 3* mootoril põhinevate mänguserverite majutust mis on tasuta, avatud lähtekoodiga ning ehitatud objektorienteeritult. Töö tulemus võiks pakkuda huvi mänguserverite majutajatele, kes soovivad lihtsustada kogu mänguserveri majutuse protsessi.

Töö tulemusena valmis veebirakendus, mis hõlmab serveri paigaldust, eemaldust, seiret, kaugjuhtimiskonsooli, FTP liidest ning mis toimib erinevate suurustega ekraanidel.

Töös kirjeldatakse rakenduse ülesehitust, kasutatud tehnoloogiaid ning integreeritavust väliste süsteemidega. Lähemalt vaadeldakse mõningate probleemide lahendamist. Töös seatud eesmärgid said täidetud ja töö tulemusena valmis reaalselt töötav rakendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 34 leheküljel, 9 peatükki, 8 joonist, 2 tabelit, 5 koodinäidet.

Abstract

Development of a web application for hosting gameservers based on *Quake* 3 engine on Linux servers

The aim of the thesis was to develop an application, which handles the hosting of game servers of games based on *Quake* 3 engines. The goal was to make it free for use, open-source and built using object-oriented programming design. The solution could be interesting for game server hosters, who want to simplify the whole game server hosting process.

The outcome of the thesis is a web-based application, which handles the server installation, deletion, monitoring, remote console access, FTP interface and which is also built with responsive web design principles, so it works on different screen sizes.

The thesis describes the structure of the application, the technologies which were used and integrations with external systems. Some of the technical problems are broken down into details. The goals set for the application were fulfilled and a real, working application was the outcome of the thesis.

The thesis is in Estonian and contains 34 pages of text, 9 chapters, 8 figures, 2 tables, 5 code examples.

Lühendite ja mõistete sõnastik

JSON	*JavaScript Object Notation* on lihtsustatud andmevahetusevorming, mis põhineb *JavaScripti* programmeerimiskeele alamhulgal. *JSON* on tekstvormingus ja programmeerimiskeelest sõltumatu. Alternatiiv *XML*-ile.
AJAX	*Asynchronous JavaScript* And *XML* on kogum omavahel seotud veebiarenduse tehnikaid, mis on kasutuses rakenduse kliendi poolel, et luua interaktiivseid veebirakendusi. *AJAX* võimaldab veebilehtedel *JavaScriptiga* tagaplaanil päringuid teha ja andmeid vastu võtta, segamata avatud lehekülje kuvamist ja olekut
FTP	*File Transfer Protocol* ehk failiedastusprotokoll on standartne arvutivõrgu protokoll, mida kasutatakse failide vahetamiseks ja muutmiseks võrgus.
RCON	*Remote Console* ehk kaugjuhtimiskonsool on protokoll, mis laseb serveri administraatoritel saata käsklusi *Quake 3* mootoril jooksvate mängude serveritele.
Quake	Arvutimäng, mis esmaselt tuli välja aastal 1996. Autori poolt loodud rakendus põhineb *Quake 3* mängumootoril
RWD	*Responsive Web Design* ehk suurustundlik veebidisain on lähenemisviis veebi disainile, mille tulemusena ei olene veebileht seadmest ega seadme ekraanisuurusest ning veebilehe kasutus ja välimus on sobilik seadmele
GET	*GET* on HTTP pärimismeetod, kus pärimisväljad on aadressireal nähtaval. *GET*-i kasutatakse üldjuhul andmete kättesaamiseks
POST	*POST* on *HTTP* pärimismeetod, kus pärimisväljad on *HTTP* päringu sees. POST-iga üldjuhul edastatakse andmeid, millega midagi ka tehakse, mitte nagu *GET*-iga, kus andmeid ainult päritakse
API	*Application Programming Interface* ehk rakendusliides on reeglistik programmiga suhtlemiseks. Kasutatakse selleks, et rakendusele saaks saata päringuid nii, et rakenduses otseselt sees olema ei peaks.
password_hash()	Tegu on funktsiooniga, mis tekitab paroolist räsi. Uuematel *PHP* versioonidel laialdasemalt kasutusel, kuna see on hetkel üks turvalisemaid meetodeid paroolide räsimiseks.
GitHub	*GitHub* on veebipõhine versioonihalduse tarkvara, põhineb *Git*-il
require_once	Pärib *PHP* faili sisse faili, näiteks klassifaili, *HTML* faili.

SSH	*Secure Shell* ehk turvakest on võrguprotokoll, mis võimaldab andmeid saata üle turvalise kanali. Põhiliselt kasutatakse *SSH*-d võrgumasinasse logimisel autentimiseks.
CSS	*Cascading Style Sheets* ehk kaskaadlaadistik on keel, milles märgitakse üles *XML* ja *SGML*- keelsete failide kujundust.
JavaScript	*JavaScript* on objektorienteeritud programmeerimiskeel, mida kasutatakse peamiselt veebilehtede skriptimiseks

Sisukord

Jooniste loetelu

Tabelite loetelu

Koodinäidete loetelu

1 Sissejuhatus

Käesolevas bakalaureusetöös ehitati avatud lähtekoodiga veebipõhine rakendus *Quake 3* mootoril jooksvate mängude serverite majutamiseks. Rakendus pakub huvi kõigile *Quake 3*-tüüpi mootoril jooksvate mänguserverite majutajatele. Varasemalt tehtud rakendustel on palju puuduseid, mida uuel süsteemil välditi. Teema valimise põhjuseks on töö autori isiklik huvi uuema põlvkonna tehnoloogia ja lahenduste vastu. Töö autor on ise ühe *Quake 3* mootoril jooksva mänguserveri majutaja ning kuna ei eksisteerinud ühtegi head lahendust mänguserverite majutamiseks, tekkis vajadus tervikliku ja parema lahenduse järele.

1.1 Eesmärgid

Töö peamiseks eesmärgiks oli luua rakendus, mis teeks uute serverite loomise ning nende haldamise võimalikult lihtsaks nii, et ainsateks ülesanneteks, mis rakendusevälised jäävad, oleks soovitava mängu, rakenduse, veebimootori, *MySQL-i* [1] andmebaasi ning *PHP* [2] interpreteerija koos vajalike laiendustega paigaldamine ja seadistamine. Muu ülesseadmine ning konfiguratsioon toimiks kõik rakenduses endas.

Teiseks eesmärgiks oli luua rakendus, mis on tasuta, avatud lähtekoodiga ning ehitatud objektorienteeritult. Bakalaureusetööga valminud rakendusel on ka sarnane rakendus hetkel olemas, kuid see rakendus ei ole avatud lähtekoodiga ja on tasuline.

Kolmandaks eesmärgiks oli ehitada rakendus kasutades *RWD* [3] printsiipe, ehk rakendus oleks kasutatava seadme ekraani suhtes suurustundlik ning kohaneks vastavalt ekraani suurusele, see on üks puudus olemasoleval rakendusel.

Viimaseks eesmärgiks oli lisada rakendusele serverite seire, mis tuvastaks, kas serveriga on hetkel probleeme. Konkureerival rakendusel oli see olemas, kuid see ei toiminud.

Rakenduse arenduseks kasutas töö autor olemasolevat virtuaalserverit ning *NetBeans* -i [4] tarkvaraarendusplatvormi. *NetBeans* -i kasuks osutus valik seetõttu, et

NetBeans toetab palju arenduskeeli (*C, Java, PHP, HTML, CSS, JS*) ning *NetBeans* -il on olemas funktsionaalsus automaatselt salvestada muudetud failid välisesse serverisse iga salvestuskäsu korral, mistõttu iga koodi rida oli koheselt saadaval ka virtuaalserveris.

2 Funktsionaalsed ja funktsioonivälised nõuded

Funktsionaalsed nõuded

- Kasutajakontod - õigused, mitmed tasemed, integratsioon foorumitarkvaradega.
- Majutusserverid - ühendus, kasutajate loomine/muutmine/eemaldamine, mänguserveri seire.
- Mänguserverid - installeerimine, esmane konfiguratsioon, mänguserveri seire.
- *API* - Veebipõhine *FTP* liides kasutajate õigustepõhised ligipääsud, üleslaadimine, allalaadimine, tekstifailide redigeerimine.
- Veebipõhise kaugjuhtimise (*RCON*) tööriist.
- Probleemide registreerimine - tickets süsteem.
- *AJAX* - enamik *GET/POST* päringud käivad läbi *AJAX'l*.
- Teavitused läbi meili.

Funktsioonivälised nõuded

- Porditavus - rakendus ei sõltu serverist.
- Laiendatavus - rakendust on võimalik liidestada teiste süsteemida läbi *API*.
- Kasutatavus - rakendust saab kasutada kõigil seadmetel nii, et rakendust ei peaks vaateid suurendama.
- Töökindlus - rakendus peab toimima ka siis, kui üks serveritest ei toimi.
- Rakendus on vabavaraline.
- Rakendus on avatud lähtekoodiga.
- Rakendus on ehitatud objektorienteeritud põhimõtetega.
- Rakenduse funktsioonid on kommenteeritud lähtekoodis.

3 Ülevaade kasutatud tehnoloogiatest

Antud bakalaureusetöö raames kasutas autor *PHP*-d serveri-poolse tarkvarana, *MySQL-i* andmebaasimootorina, *jQuery*-t [5] *JavaScripti* käskude lihtsustamiseks ning lühendamiseks, *BootStrap-i* [6] ning *Angle Bootstrap Admin App-i* [7] visuaalseks pooleks, *HTML-i* [8] kasutajaliidese ehituseks, *jQuery* dataTables-it [9] erinevate suurtemate andmete haldamiseks ning kuvamiseks, Select2.js-i [10], et kasutada otsingut *HTML* select objektidel ning toastr-it [11], et kuvada teavitusi kasutajale. Järgnevalt on lahti seletatud erinevad tehnoloogiad.

3.1 PHP ja MySQL

Rakenduse serveripoolse osa arendamiseks kasutas töö autor *PHP*-d. *PHP*, ehk *PHP*: *Hypertext Preprocessor* on laialdaselt kasutusel olev avatud lähtekoodiga programmeerimiskeel, mis on sobilik veebiprogrammide ehitamiseks ning mida saab kasutada ka *HTML-i* vahel. *PHP* kood jooksutatakse serveris, mis genereerib sellele vastavale *HTML-i* ning kuvab seda kasutajale. *PHP* on aktiivses arenduses, kirjutamise hetkel on kõige uuem versioon 7.1. Autori rakendus baseerus *PHP* 7.0 versioonil [2]. *MySQL* on avatud lähtekoodiga relatsiooniline andmebaasi haldussüsteem. [1] *MySQL* on üks populaarseim andmebaasitarkvara, seda kasutavad erinevad foorumitarkvarad, näiteks *IP.board, phpBB, MyBB* ning suuremad veebilehed, näiteks Google (mitte otsingute jaoks), Facebook, Twitter ning Youtube [12]. Otsus kasutada *PHP*-d ja *MySQL*-i, tekkis tänu töö autori uuringu tulemusena, millest selgus, et pea kõik, kes majutavad ise foorumi/veebitarkvara, kasutavad selleks ka *PHP*-d ja *MySQL*-i. Samuti on *PHP* ja *MySQL* kasutusel seetõttu, et oleks võimalik ehitada liides, mis võimaldab modernsetes foorumitarkvarades kasutada foorumikontosid otse veebirakenduses, kuna modernsemad foorumitarkvarad kasutavad *PHP* ja *MySQL*-i. Vanematel foorumitarkvaradel on oma lahendused paroolide räsimiseks, aga uue *PHP* versiooniga (nimelt *PHP* 5.5-ga) tuli kaasa universaalsem lahendus paroolide räsimiseks, milleks on funktsioon *password_hash()* [13], *password_hash()* funktsiooni saab ära kasutada foorumiga autentimiseks.

3.2 jQuery

jQuery on *JavaScripti* raamistik, mis lihtsustab *JavaScripti* koodi kirjutamist ning muudab loodava lehekülje dünaamilisemaks ning muudab *HTML*i dokumendi käsitlemise *JavaScripti*s ja AJAX päringud lihtsamaks [5]. *jQuery* on avatud lähtekoodiga, väikese suurusega ning kiire. Samuti on *jQuery* üks kõige populaarsem ning enim kasutatav *JavaScripti* raamistik [41]. *jQuery*-t arendatakse aktiivselt ning kirjutamise hetkel on viimane versioon versioon 3.2.1 [14], mida kasutas ka autor oma töös. Tänu varasemale kokkupuutele *jQuery*-ga otsustas töö autor kasutada just seda raamistikku kuna see lihtsustab *JavaScripti* programmeerimist.

3.3 Bootstrap ja Angle Bootstrap Admin App

Bootstrap ja Angle Bootstrap Admin App

Bootstrap on *HTML*, CSS ja JavaScript raamistik, millega saab ehitada suurustundlike lehekülgi, mis näevad telefonis head välja [6]. Angle Bootstrap Admin App on laiendus BootStrap-ile, mis on loodud, et leheküljele anda administratiivse paneeli väljanägemine [7]. Angle Bootstrap Admin App ning BootStrap-i kasuks otsustas autor, kuna see on sobilik autori rakendusele.

3.4 HTML

HTML, ehk Hypertext Markup Language on standardne märgenduskeel veebilehtede ning veebirakenduste loomiseks. Koos *CSSi* ning *JavaScripti*ga, loob *HTML* triaadi tehnoloogiateks, mida kasutatakse *World Wide Web-is*. *HTML* kirjeldab veebilehe struktuuri ning omab veebilehe sisu [8].

3.5 jQuery Datatables

jQuery DataTables on plugin *jQuery*-le, mis muudab tavapärase *HTML* tabeli rohkem interaktiivsemaks. *jQuery* DataTables omab otsingufunktsiooni, leheküljestamist ja sorteerimist ilma, et arendaja peaks need funktsionaalsused ise välja töötama [9]. Töö autor otsustas *jQuery* DataTables kasuks kuna *jQuery* DataTables-iga saab andmeid laadida sisse *AJAX-iga* ning ka automaatse leheküljestamise tõttu. Kuna autori

rakenduses on nähtaval ka erinevad logid erinevatel funktsionaalsustel, võivad need logid väga ruttu muutuda suureks, mistõttu oleks pidanud välja töötama leheküljestamise. Kuid selle asemel, et arendada funktsioon, mis ise tegeleb leheküljestamise poolega, on *jQuery* DataTables selleks piisav lahendus.

3.6 Select2.js

Select2.js on *jQuery* asendus *HTML* select objektile. Select2.js on hea vahend siis, kui on soov kasutada valikuobjekti *HTML*-is, aga valikuid sinna objekti tekiks liiga palju, mistõttu nendest läbi kerimine oleks otstarbetu. Select2.js-il on olemas otsingufunktsionaalsus, mille saab ühendada erinevate andmeallikatega (*JSON*, *AJAX* ja palju muud) [10]. Autor otsustas Select2.js antud töös on kasutatudseda peamiselt selle pärast, et väliseid kasutajakontod, mida saab ühendada rakendusega, võib olla palju, siis saab kasutajaid lihtsalt välisest andmebaasist otsida.

3.7 Toastr

Toastr on *JavaScripti*l ning *jQuery*-l põhinev laiendus, millega saab lihtsalt kuvada erinevaid teavitusi kasutaja ekraanil, kuid mis ei takista lehekülje edasikasutamist [11]. *Toastr-it* on väga mugav kasutada ning erinevaid teateid läbi selle saata, seega töö autor kasutas seda lahendust.

4 Rakenduse kirjeldus

Q3Panel veebirakendus on suunatud *Quake 3* serverite majutajatele, kes soovivad lihtsustada serverimajutusprotsessi ning jätta kogu serverihaldus serverirakenduse kätte. Põhiliseks eesmärgiks antud rakenduse puhul on kaotada probleemid, mis on seotud serverimajutusega, tuues näiteks automaatsed taaskäivitused ja ka mitmete serverite jooksutamine ühel majutusserveril.

Rakendus on jaotatud 13-sse vaatesse, millest igal ühel on oma funktsionaalsus, enamikel oma klass. Rakendus on avatud lähtekoodiga, programmeeritud objekt-orienteeritult. Rakenduse arendades jälgiti arendamise häid tavsid ning tükeldati kood erinevateks tükkideks, millest suuremad tehti klassideks. Kõik lähtekoodi funktsioonid on kommenteeritud ning erinevatesse failidesse laiali hajutatud, et lähtekood oleks loetavam. Lähtekood ning kommentaarid on inglise keeles.

Rakenduse back-end on ehitatud *PHP*'s [2] ja andmebaasitarkvaraks *MySQL* [1]. Front-end on ehitatud *HTML*[8], *CSS*[15] ja JavaScript-is[16] . Rakenduses on jälgitud *RWD* [3] printsiipe, mistõttu on rakenduse vaade sobilik ka telefonis ilma, et peaks vaateid suurendama/vähendama. Rakendus on võimalik ühendada välise andmebaasiga, näiteks foorumitarkvaraga, mille parooliräsimine toimib *PHP* funktsiooni *password_hash()* [13] baasil.

Rakendusele on ehitatud ka meilimisfunktsioon *PHP*Mailer[37] ning SendGrid[38] baasil. SendGrid on pilvepõhine *SMTP*[39] teenuse pakkuja, mis laseb meile saata meiliserverit omamata. SendGrid-il on mitmed teenused ning pakkumised, millest üks on ka tasuta teenus [39]. *PHPMailer* on *PHP*-l põhinev laiendus, millega saab lihtsalt saata meile. *PHPMailer* on üks populaarseim lahendus *PHP*-s meilide saatmiseks. [40] Rakenduses saab rakenduse administraator ise vastavalt oma eelistusele valida, millist meetodit ta soovib meilide saatmiseks.

Erinevad failid on jaotatud erinevatesse kaustadesse vastavalt nende tüübile. *Javascript*, *CSS* ning staatiline *HTML* osa on jaotatud *static* kausta, klassid *classes* kausta ning erinevad vaated jaotatud vaadete kausta.

5 Arendusprotsess

Rakenduse *Q3Panel* arendamist alustas autor esmalt arendusplaani välja töötamisega. Autor pani kirja erinevad funktsionaalsused mis võiksid olla rakendusel olemas, analüüsis sarnase rakenduse eeliseid ning puudusi ja töötas välja selle põhjal funktsionaalsuse plaani. Järgnevalt uuris autor kuidas servereid paigaldada, eemaldada, kuidas uusi serverikasutajaid luua, kuidas serverit käivitada ja kuidas seda kontrollida. Peale väljatoodud probleemide lahendamist algas arendusprotsess.

Arendusprotsess esimene etapp oli *GitHubi* [17] versioonihaldussüsteemi salve loomine, et oleks ülevaade erinevatest sooritustest ja funktsionaalsustest, mis on tehtud, mis on pooleli ning samuti, et lähtekood oleks kättesaadaval kõikjalt ning arendusprotsessi sammud oleksid läbipaistvad. Lõplik versioon koosneb 92-st üleslaetud sooritusest. Arendusprotsessi teine etapp oli üles ehitada andmebaasiga suhtlemise funktsionaalsus, andmebaasi tabelite definitsioon ja paigalduslehted. Pärast seda arendati välja funktsionaalsus vaatepõhiselt.

5.1 Vaated enne rakenduse paigaldust

Vaadete funktsioonide poolel olevad tähistused:

- Klass::Funktsioon märgib ära, et tegu on staatilise funktsiooniga, milleks ei ole vaja konstrueerida objekti.
- Klass->Funktsioon märgib ära, et tegu on objekti-põhise funktsiooniga, milleks peab olema eelnevalt konstrueeritud objekt.
- *Fail.PHP* - Funktsioon märgib ära, et tegu on failipõhise funktsiooniga, millel pole eraldi klassi ehitatud.

Paigaldusvaade. Kuna üks eesmärkidest oli, et rakendus oleks hallatav ja lihtsasti kasutatav, siis oli vajalik ka paigalduslehed luua. Paigaldusleht kogub kokku info, mida on vaja rakenduse edukaks töötamiseks. Paigalduslehed pärivad kasutajalt andmebaasi infot, valikul ka välise süsteemi andmebaasi infot, et oleks võimalik kasutada välise andmebaasi kasutajaid, meilisüsteemi konfiguratsioon ning paigaldusleht käsitleb ka esmase kasutajakonto loomist. Loogikat juhib Installation klass.

Funktsioonid antud vaate jaoks:

- *Installation initializeConfig* – initialiseerib *config.php* faili, kus seisneb rakenduse aadress ning andmebaasi informatsioon.
- *Installation initializeTables* – loob tabelid andmebaasi rakenduse kasutamiseks.
- *Installation::initializeExternalConnection* – initialiseerib välise andmebaasi ühenduse ning kirjutab välise andmebaasi informatsiooni lokaalsesse andmebaasi.
- Email::*saveEmailPreferences* – salvestab meilinduse seaded andmebaasi.
- *User::addUser* – lisab uue kasutajakonto.

5.2 Vaated peale rakenduse paigaldust

5.2.1 Üldine

Kõigil rakenduse lehtedel on kontroll, kas kasutaja on sisse logitud ja kas tal on piisavalt õigusi, et lehel olla. Õiguste kontroll on juhuks, kui keegi on muutnud kasutaja õigusi peale seda, kui kasutaja on sisse loginud. Sisselogimise vaate pärimine on *PHP require_once* [18] funktsiooniga esile kutsutud ning päringud saadetakse klassi Login. Sisselogimise vaate pärimise eelis on see, et kui kasutaja proovib spetsiifilisele aadressile siseneda, siis teda mitte ei suunata eraldi lehele, kus võib tekkida aadressivahetus, vaid kõik vajalik päritakse sisse samale aadressile, kuhu sooviti siseneda. Kui kasutajal on õigused rakendusel olemas, siis järgnev üldine vaade on küljepaneel, kus on olemas kõik lingid erinevatele vaadetele. Kuna tegu on staatilise informatsiooniga, siis leidis autor, et külgmise paneeli lisamine igale lehele eraldi, teeb lähtekoodi vähem loetavamaks ning otstarbetult suureks. Samuti on ka erinevate teemade valik lisatud staatilisele lehele. Erinevad teemad muudavad värviprofiile rakenduses vastavalt kasutaja soovile. Teemade eelistused salvestatakse lokaalsesse andmebaasi. Kõik päringud rakendusel suunatakse lehele *functions.php* või samale lehele, kus hetkel kasutaja viibib. Kuna funktsioonide fail päritakse sisse igal lehel, on võimalik saata ka päringuid otse lehele, mitte täpselt funktsioonide failile. Funktsioonide failis esinevad osad funktsioonid mille kohta autor leidis, et ei ole otstarbekas luua uut klassi ning samuti esineb funktsioonide failis kõikide *HTTP*

[20] *POST* [19] ja *GET* [20] käskude kuulamine, klasside konstruktsioon vastavalt käsule ning vastava info väljastamine.

Funktsioonid üldisel vaatel:

- U*ser::checkUser* – kontrollib, kas antud kasutajakontol on samad parameetrid mis olid ka sisselogimisel.
- *isset* – kontrollib, kas arendaja poolt kontrollitud parameetrid on sätestatud.
- *User->authenticate* – autendib kasutajakonto rakendusega.
- *User::forgotPassword* – salvestab andmebaasi unustatud parooli võtme ning saadab e-maili kasutajale, kes seda taotles. Ei toimi siis, kui kasutajakonto tuleneb välisest andmebaasist.
- *User::recovery* – kontrollib unustatud parooli võtit, kas kasutajale võib unustatud parooli vormi näidata.
- *User::changeForgottenPassword* – vahetab unustatud parooli kasutaja sisestatud parooliks juhul, kui võti, millega paroolitaastamist päriti, vastab andmebaasis olevale võtmele.
- *functions.php – generateRandomKey* – genereerib juhusliku võtme. Sisselogimisvaates genereerib juhusliku võtme unustatud parooli jaoks.
- *User::changeUserStylePreference* – salvestab andmebaasi kasutaja eelistuse värviteemale.
- *User::setSessionVariables* – seab sessiooniparameetrid sisseloginud kasutajale.

5.2.2 Pealehe vaade

Pealehe vaatel on kasutajat tervitav tekst, paar märksõna mida telefonivaates teha ning rakenduse statistika: mitu kasutajakontot on rakenduses, mitu töötavat serverit, kas väline andmebaas on ühendatud ja millist meilinguteenust kasutatakse. Loogika juhtimist pealehel pole.

5.2.3 Mänguserverite vaade

Mänguserverite vaatel on ülevaade olemasolevatest serveritest ja nende andmetest. Vastavalt kasutaja õigustele kuvatakse informatsioon, mida kasutaja näha võib.

Mänguserverite vaates saab siseneda serveripõhisesse vaatesse ning ka lisada uusi servereid. Mänguserverite nimekiri pärineb Server klassist

Funktsioonid mänguserverite vaatel:

- *User::canPerformAction* – funktsioon, mis kontrollib andmebaasist kasutaja hetkelist gruppi vastavalt etteantud grupile. Kui etteantud grupp on sama või väiksem, kui kasutaja grupp, tagastatakse true, vastaval juhul false.
- *Server::getServersWithHostAndGame* – tagastab serverid koos majutusserverite ning mängudega. Serverite nimekiri on limiteeritud vastavalt kasutaja vastendusele.
- *Host::getHostsSelect* – tagastab *HTML* rippmenüülisti majutusserveritest.
- *Game::getGamesSelect* – tagastab *HTML* rippmenüülisti olemasolevatest mängudest.
- *Server->addServer* – lisab uue mänguserveri.

5.2.4 Serveripõhine vaade

Serveripõhine vaade on spetsiifilisem vaade mänguserveri vaatele. Töö autor leidis, et kui kogu informatsioon ning funktsionaalsus panna mänguserveri vaatele, muutub mänguserverivaade liiga suureks, mistõttu eraldati mänguserveri ja serveripõhine vaade. Serveripõhises vaates, vastavalt õigustele, saab serverit muuta, keelata serveri kasutust, kustutada, tööle panna või sulgeda ning veebipõhise kaugjuhitavusesse pääseda. Kõigi funktsionaalsuste loogika asub Server klassis. Veebipõhine kaugjuhitavus on liides, mis suudab suhelda *Quake 3* mängu serveritega vastavalt *Quake 3* standardile [21]. Serveripõhises vaates saab ka muuta serverikonto parooli, kas ise uue parooli sisestades või lastes rakendusel genereerida uus parool. Samuti on järgnevad lingid serveripõhises vaates kasutajate vastendamiseks ning ka veebipõhise *FTP* liidesele.

Funktsioonid serveripõhises vaates:

- *Server->disableServer* – sulgeb serveri ning keelab selle, mistõttu seda enne taaslubamist kasutada ei saa. Käskluse saab saata ainult rakenduse administraator.
- *Server->enableServer* – annab loa serveri käivitamiseks ning haldamiseks. Käskluse saab saata ainult rakenduse administraator.

- *Server->updateServer* – uuendab serveri informatsiooni vastavalt kasutaja sisestatule.
- *Server->startServer* – käivitab serveri
- Server->stopServer – peatab serveri.
- *Server->sendQ3Command* – saadab serverile käskluse vastavalt *Quake 3* standardile. Kutsutakse välja serveri automaatse kontrolli käigus ning ka veebipõhise kaugjuhtimise käigus.
- *Server::getServersWithHostAndGame* – koos serveri identifikaatoriga, tagastab antud serveri informatsiooni, ilma serveri identifikaatorita tagastab kõik serverid.
- *Server->changeServerAccountPassword* – muudab serveri konto parooli majutusserveris.
- *Server->deleteServer* – kustutab serveri majutusserverist ning rakendusest. Käskluse saab saata ainult rakenduse administraator.

5.2.5 Kasutajate vastendamise vaade

Kasutajate vastendamise vaates saab vastendada erinevaid kasutajaid serverile. Vastendamise tähtsus tulenes arendusplaani ning sarnase rakenduse analüüsi käigus. Sarnasel rakendusel olid serverid vastendatud üks-mitu suhtega, ehk igal tavakasutajal saab olla mitu mänguserverit, aga mitmel kasutajal ei saa olla üks mänguserver, kuid see muutis kogu serverihalduse suhteliselt tülikaks, mistõttu leidis autor, et oma rakenduses soovib ta vastendada serverid kasutajate vahel mitu-mitu suhtega, ehk ühel serveril võib olla vastendatud mitu kasutajat ning ühel kasutajal võib olla vastendatud mitu serverit. Vastendamisel seatakse ka kasutaja õigused. Kui kasutaja grupp rakenduses on serveri omanik, siis on tal automaatselt vastendatud serverile kõik õigused. Kui kasutaja grupp rakenduses on rakenduse administraator, siis on kõik mänguserverid temale vastendatud. Kui kasutaja grupp rakenduses on tavakasutaja, võetakse kasutaja õigused vastavalt seatule. Vastendamise vaates saab vastendada uusi kasutajaid, muuta ning kustutada olemasolevaid vastendusi. Loogikat juhib Server klass.

Funktsioonid kasutajate vastendamise vaates:

- Server::*addUserMapping* – vastendab antud kasutaja serverile.
- Server::*editUserMapping* – muudab kasutaja vastendust serverile.

- Server::*removeUserFromMapping* – eemaldab kasutaja vastenduse vastavalt serverilt.

5.2.6 Veebipõhine FTP liides

Kuna leidub kasutajaid, kellel puudub varasem kogemus *FTP*-ga, oli mõttekas teha rakendusele veebipõhine *FTP* liides. Veebipõhist *FTP* liidest juhib JavaScript ja *FTP* klass, kus *FTP* klass sõltub Serveri objektist. Veebipõhisele *FTP* liidesele on lisatud lihtsamad funktsionaalsused, nagu uue tekstifaili loomine, uue kausta loomine, uue faili üleslaadimine, faili kustutamine, ümber nimetamine, muutmine ning allalaadimine. Rakendus käivitab faili allalaadimise juhul, kui faili ei saa tekstipõhiselt muuta. Kui fail on muudetav, avab liides modal-akna koos faili sisuga. Veebipõhine *FTP* liidese *JavaScripti* pool tegeleb *FTP* tabeli initialiseerimisega ning failide/kaustade käskude edastamine *back-end'i*.

Funktsioonid veebipõhise *FTP* liidese vaates:

- *FTP->getDirectoryFileList* – tagastab vastava kausta sisu.
- *FTP->fpt_is_dir* – kontrollib, kas antud asukoht on kaust või fail.
- *FTP->getFileContents* – tagastab faili sisu kasutajale.
- *FTP->getFileDownloadURI* – tagastab *FTP* lingi vastavale failile, et see alla laadida.
- *FTP->renameFileOrFolder* – muudab faili või kausta nime.
- *FTP->createNewFolder* – lisab uue kausta.
- *FTP->createNewFile* – lisab uue faili koos kasutaja sisestatud sisuga.
- *FTP->deleteFileOrDir* – eemaldab kausta/faili. Kui tegu on kaustaga, siis kustutab kausta sisu rekursiivselt ning viimase sammuga kustutab kausta.
- *FTP->uploadNewFile* – laeb uue faili serverisse. Fail laetakse ülesse kasutaja poolt ja *uploadNewFile* funktsioon võtab selle faili sisu ning laeb selle ülesse serverisse.
- *FTP->writeFile* – faili muutmisel välja kutsutav funktsioon. Kustutab olemasoleva faili ning kirjutab uue faili koos uue sisuga.

5.2.7 Majutusserveri vaade

Kuna mänguserveritel peab olema ka majutusserver, oli vaja ehitada eraldi tabel majutusserverite jaoks koos vaatega, et mänguserverid oleks mitu:üks suhtega majutusserveritega. See muudab serverite paigaldamise lihtsamaks, ei pea iga uue serveri sisestamisel uuesti majutusserveri andmeid sisestama. Majutusserveri vaatele on ligipääs ainult rakenduse administraatoritel. Majutusserveri vaates saab muuta olemasolevaid majutusservereid ning lisada uusi majutusservereid. Loogikat juhib Host klass.

Funktsioonid majutusserveri vaates:

- *Host->updateHost* – uuendab majutusserveri andmeid.
- *Host->deleteHost* – teostab kontrolli, kas majutusserveril on ka mänguservereid, kui ei ole, siis eemaldab majutusserveri rakendusest.
- *Host::getHosts* – tagastab majutusserverid vastavalt väljakutsele, kas spetsiifilise majutusserveri ja/või kas lisada ka majutusserveri parool väljavõttesse.
- *Host->addHost* – lisab uue majutusserveri rakendusele. Teostab ka kontrolli majutusserveris, saates käsu *whoami* [22] üle *SSH* [23] ühenduse, et kontrollida, kas rakendus saab ühendust majutusserveriga.

5.2.8 Kasutajatoe piletite vaade

Kasutajatoe piletites saavad kasutajad teha erinevaid pileteid rakenduse administraatoritele erinevatest probleemidest. Piletitel registreeritakse kasutaja sisestatud info ning sisselogitud kasutaja info. Kasutajatoe piletid hoiavad samuti ka ajalugu erinevate piletite üle. Piletitel on 4 staatust: avatud, suletud, ei saanud probleemi uuesti esile kutsuda ning lahendatud. Kasutajatoe piletite vajadus tekkis sellest, et oleks võimalik hoida probleemide ajalugu ning nende lahendused rakenduse lehel. Loogikat juhib *Ticket* klass.

Funktsioonid kasutajatoe piletite vaates:

- *Ticket->saveTicket* – salvestab uue kasutajatoe pileti andmebaasi.
- *Ticket->mapTicketToUsers* – vastendab kasutajatoe pileti vastavatele kasutajatele olenevalt funktsiooni väljakutsest.

- *Ticket->saveMessageToTicket* – salvestab uue kasutajatoe pileti sõnumi andmebaasi.
- *Ticket->setStatus* – salvestab kasutajatoe piletile uue staatuse.
- *Ticket->notifyMappedUsers* – saadab e-maili kasutajatoe piletiga vastendatud kasutajadele.
- *Ticket->isUserMappedToTicket* – tagastab *true*, kui kasutaja on vastendatud kasutajatoe piletiga, vastasel juhul f*alse*.
- *Ticket::getTicket* – tagastab kasutajatoe pileti vastavalt identifikaatorile.
- *Ticket::getTickets* – tagastab kasutajatoe piletid.

5.2.9 Kasutajate vaade

Kasutajate vaatel saab oma kasutajakontot muuta ning vastavalt õigustele ka teiste kasutajakontosi muuta ning lisada. Kui rakendusega on ühendatud väline andmebaas, saab lisada ka väliseid kasutajakontosi kasutajateks rakendusse. Välistel kasutajakontodel salvestatakse välise andmebaasi kasutajakonto identifikaatorväli lokaalse andmebaasi kasutajanime välja ning seatakse lähteväärtus *(origin)* 1'ks, et rakendus teaks, et seda kasutajakontot peaks käsitlema kui välist kasutajakontot. Kasutajade loogikat juhib *User* klass.

Funktsioonid kasutajate vaates:

- *User->register* – registreerib uue kasutaja rakendusse.
- *User::editAccount* – muudab kasutajakonto andmeid.
- *User::deleteAccount* – kustutab kasutaja rakendusest.
- *User::getUserById* – tagastab kasutajakonto vastavalt identifikaatorile.
- *User::canEditUser* – tagastab informatsiooni, kas ja mida saab vastav kasutaja vastava kasutajaga teha/muuta.
- *User::getAllUsers* – tagastab kõik rakenduse kasutajakontod. Kui on tegu ka välise andmebaasi kontoga, siis koondab informatsiooni kokku ka välisest andmebaasist.
- *User::getExternalAccountSelect2* – tagastab *Select2*-vastavale [10] standardile välise andmebaasi kasutajakontode nimekirja vastavalt kasutaja sisestusele.

5.2.10 Mängude seadistus

Mängude seadistuse vaates saab uusi mänge seadistada rakendusse ning ka olemasolevaid mänge seadistada. Mängude seadistus on vajalik, kuna rakendus peab teadma, mis faile tuleb kopeerida uue mänguserveri ülesseadmisel. Samuti on mängudel vaja ka käivituskäsklust, mis käskudega mäng läheb tööle. Mängude seadistuse loogikat juhib *Game* klass.

Funktsioonid mängude seadistuse vaates:

- *Game::getGames* – tagastab mängud vastavalt funktsiooni väljakutsele.
- *Game::saveGame* – sisestab uue mängu andmebaasi
- Game::deleteGame – kustutab mängu. Kontrollib esmalt, kas on mänguservereid, millel on see mäng hetkel olemas.
- *Game::updateGame* – uuendab andmebaasis mängu andmeid.

5.2.11 Rakenduse seadistuse vaade

Rakenduse seadistuse vaates saab muuta meiliseadistust ning välise andmebaasi ühenduse seadistust. Välise andmebaasi seadistuse loogikat juhib *functions* fail ning meiliseadistust juhib Email klass.

Funktsioonid rakenduse seadistuse vaates:

- Kuna rakenduse seadistuse vaatele ei olnud otstarbekas klassi ehitada, tegeleb funktsionaalsuse poolega *functions.php*

5.2.12 Logide vaade

Logide vaade jaguneb kolmeks. Esimeseks on rakenduse logid. Rakendus logid hoiavad informatsiooni, mis on rakenduses tehtud, näiteks erinevate kasutajate muutmine, erinevate muudatuste läbi viimine ja palju muud. Rakenduse logisi näevad rakenduse administraatorid. Teiseks on serveri logid. Serveri logid hoiavad kõike informatsiooni, mis on serveriga toimunud. Peamiselt seisneb serverilogides informatsioon serveri probleemide ja erinevate käskude kohta. Näiteks kui server peaks olema kokku jooksnud ja automaatne serveri taaskäivitaja tuvastab selle vea, logib ta selle ka serveri logidesse. Serveri logid saadakse vastavalt kasutaja õigustele ja vastendusele. Viimaseks on põrunute sisselogimiste logimine. Põrunud

sisselogimised logitakse sisselogimisel ning seal koondatakse informatsioon kontodest, millega üritati sisse logida ning IP aadressid. Põrunute sisselogimiste logi näevad ainult rakenduse administraatorid. Kogu logimise loogikat juhib Logger klass, mis juhib nii logi sisestamist andmebaasi kui ka logi kättesaamist andmebaasist. Tulemused kuvatakse *jQuery* DataTable'ga ning saadakse läbi *AJAX* kutse functions failile.

Funktsioonid logide vaates:

- *Logger::getServerLogs* – tagastab serverilogid vastavalt funktsiooni väljakutsele.
- *Logger::getFailedLogins* – tagastab põrunud sisselogimise katsed. Vaade nähtaval ainult rakenduse administraatoril.
- *Logger::getLogs* – tagastab rakenduse logi. Nähtaval ainult rakenduse administraatoril.

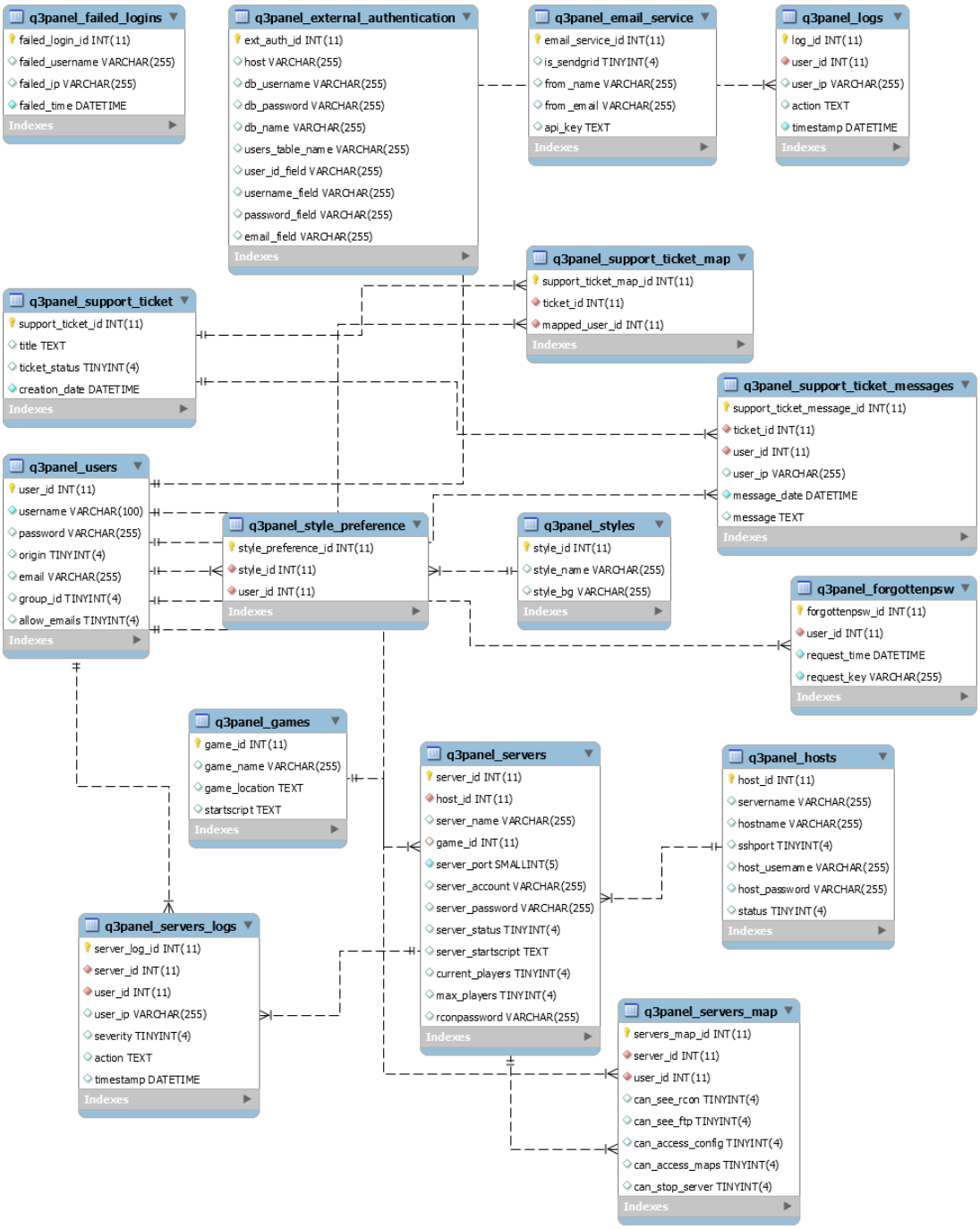
Erinevad funktsioonid mis ei ole vaate-põhised:

- *Logger::log* – logib rakenduse-põhilisi tegevusi
- *Logger::logFailedLogin* – logib põrunud sisselogimiskatse
- *Logger::logServer* – logib serveri kohta informatsiooni.
- Email*::notifyServerUsers* – teavitab e-maili teel serveriga vastendatuid kasutajaid.
- Email::*getEmailPreferences* – tagastab meili saatmise seadistuse.
- Email->sendEmail – saadab e-maili vastavalt objekti konstruktsioonile
- Server->*checkServer* – kontrollib mänguserveri hetkelist olukorda, kas server töötab, kas ta vastab *Quake 3* getinfo [21] käsule ja palju on serveris hetkel mängijaid.
- Server::g*etRunningServers* – tagastab hetkel töötavad serverid.
- *SQL* klass – Kontrollib kogu andmebaasi ja rakenduse vahelist suhtlust. Põhineb *PDO'l* aga standardiseeritud rohkem autori poolt, et erinevate käskluste jaoks ei peaks erinevaid lisafunktsioone välja kutsuma.
- *SSH->sendCommand* – saadab käsu läbi *SSH* [23] protokolli vastavalt objekti konstruktsioonile.
- *User::getExternalQuery* – muudab *SQL* päringus puuduvad väljad vastavalt funktsiooni väljakutsele vastavateks väljadeks.

- *User::getExternalConnection* – tagastab uue *SQL* objekti vastavalt funktsiooni väljakutsele.
- *Writer->write* – kirjutab faili vastavalt konstruktsioonile ning funktsiooni väljakutsele.
- *loader.php* – fail, millega päritakse sisse kõik vajalikud klassid ning failid rakenduse toimimiseks.
- *Constants.php* – hoiab konstantseid andmeid rakendusele, näiteks andmebaasi päringud, rakenduse nimi, mida kuvada tiitli väljal, erinevad staatused, *JavaScripti*/*CSS'i* sisselaadimine, serverikäsud, *SSH* käsud ja palju muud.
- *functions.php* – pea iga funktsiooni väljakutse alguspunkt. *GET/POST* päringud suunatakse siia ja vastavalt parameetritele täidetakse käsk. Hoiab samuti juhusliku võtme genereerimisfunktsiooni, täisarvu tõeväärtuseks muutmine, tõeväärtuse täisarvuks muutmine ning tõeväärtuse sõnaks muutmine.
- *q3panel.js* – Hoiab erinevaid *JavaScript'i* funktsioone, mida kutsuda välja erinevatel lehtedel. Lisab linkidele active klassi, kui kasutaja on hetkel selle lingi peal. Hoiab funktsiooni, mis käsitleb enamike vorme. Hoiab erinevate modalite [25] initialiseerimisfunktsioone. Hoiab palju *AJAX* [24] funktsioone. Hoiab erinevaid *FTP* funktsioone, mida *back-endile* edastada.

5.3 Andmebaas

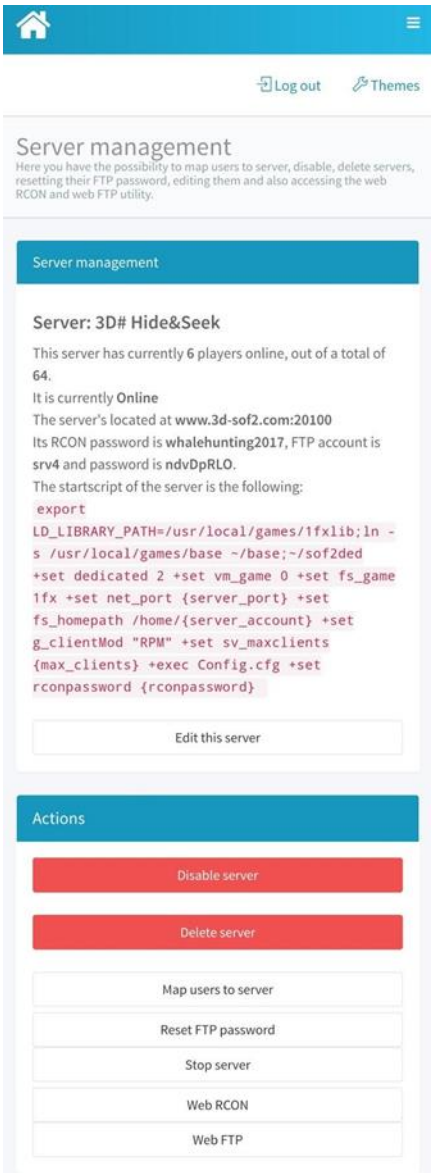
Andmebaas on üles ehitatud relatsiooniliselt ning igal tabelil on viide, vastavalt andmebaasi tabelile, ta välisvõtmetele. Andmebaasis hoiab rakendus kõiki rakenduse andmeid, väljaarvatud rakenduse aadress, andmebaasi ühenduse parameetrid ning konstantsed andmed, mis on defineeritud *Constants* klassis. Joonisel 1. leiate andmebaasi olemi-suhte skeemi.



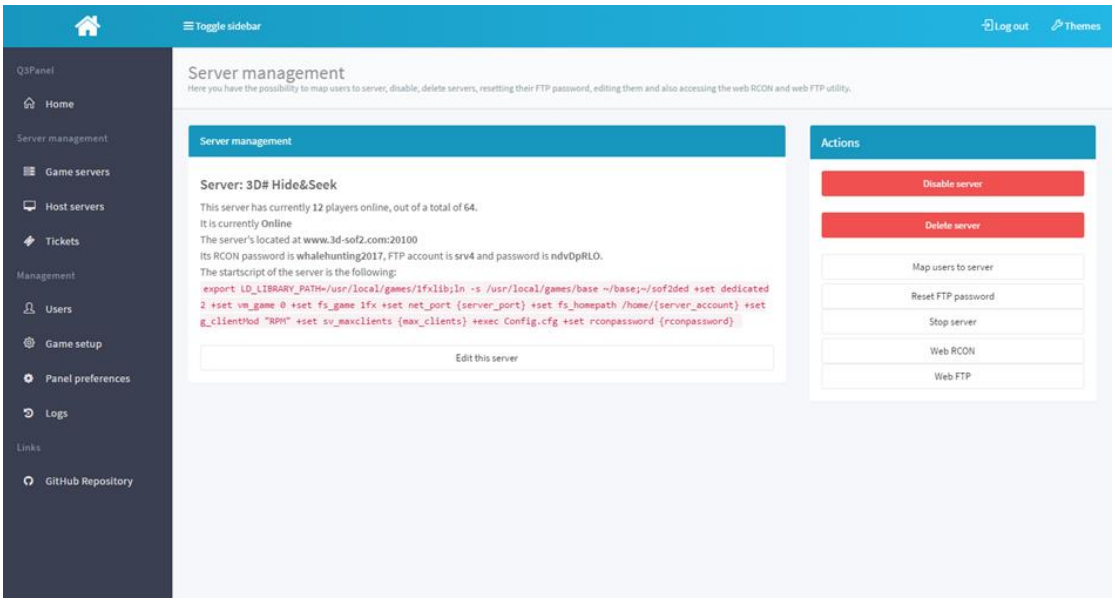
Joonis 1. Andmebaasi olemi-suhte skeem

6 Rakendus

Lõplikus rakenduses sai kõik püstitatud eesmärgid täidetud. Üks peamisi eesmärke rakenduse loomisel oli see ehitada *RWD* printsiipidega, mis ka edukalt täideti. *RWD* printsiipidel paigutatakse elemendid vastavalt ekraani laiusele. Mobiilivaates paigutatakse suuremad elemendid üksteise alla, arvutis üksteise kõrvale. Näide selleks on joonisel nr. 2 ja joonisel nr. 3, kus serveri halduse nupud on telefoni vaates lehekülje alaosas, kuid arvutivaates on serveri detailide kõrval.

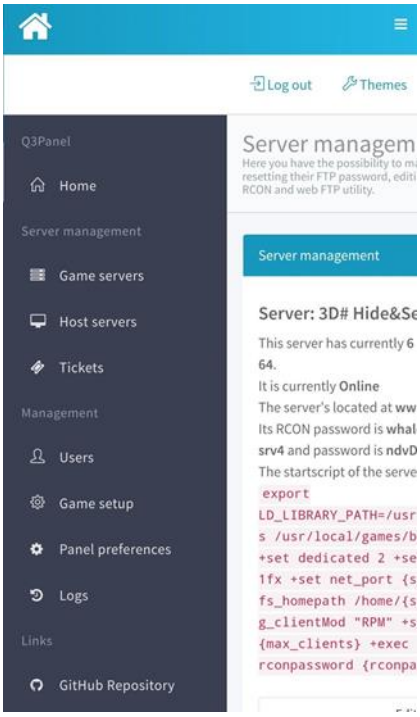


Joonis 2. Serverite detailvaade telefonis

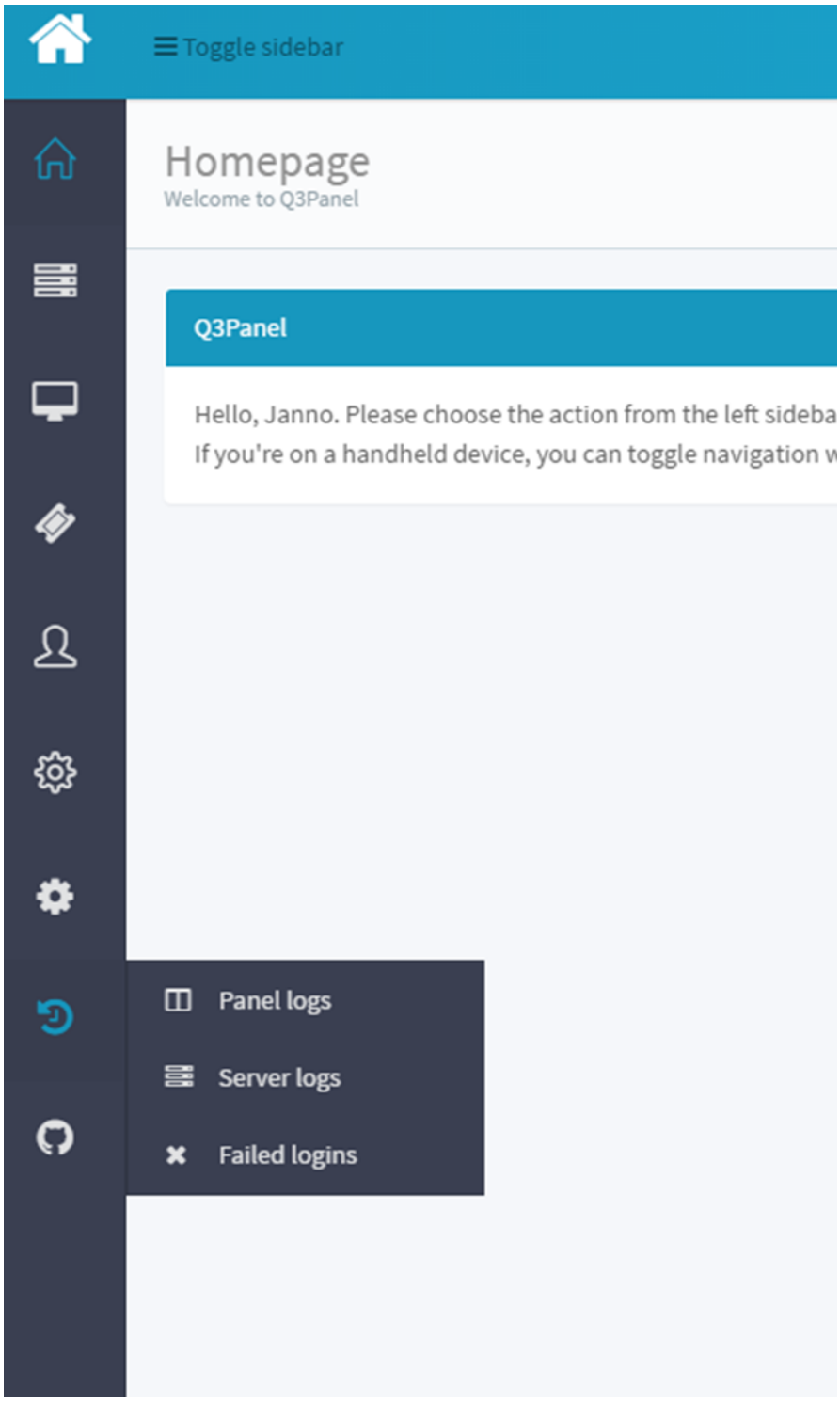


Joonis 3. Serverite detailvaade arvutis

Mobiilivaates pole näha navigeerimislinke, kuid need saab esile kutsuda vajutades vasakult nurgast nuppu. Navigeerimislingid on algselt telefonivaates peidetud ruumi kokkuhoiu tõttu. Arvutivaates on lingid nähtaval alati ning neid saab minimeerida. Laiendatud navigeerimislingid mobiilivaates on välja toodud joonisel 4 ning arvutivaates minimiseeritud navigeerimislingid on välja toodud joonisel 5.

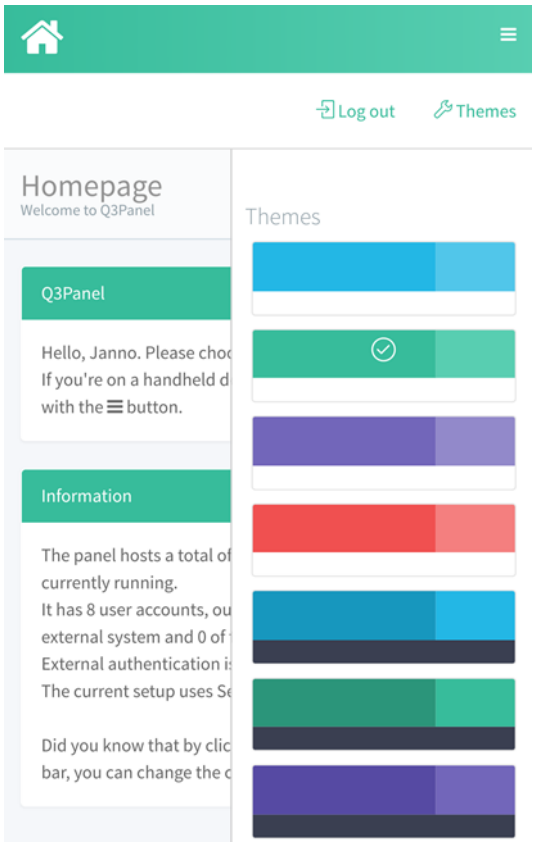


Joonis 4. Laiendatud navigeerimislingid mobiilivaates



Joonis 5. Minimeeritud navigatsioonimenüü arvuti vaates

Samuti on rakendusel olemas värviprofiili eelistused, kus hetkel on olemas 8 värviprofiili. Värviprofiilid muudavad rakenduse värve, kuid rakendus jääb ise samaks. Värviprofiilide valikumenüü ning erinevad värviprofiilid on välja toodud joonisel 6.



Joonis 6. Rakenduse värviprofiilide valik koos hetkel valitud värviprofiiliga

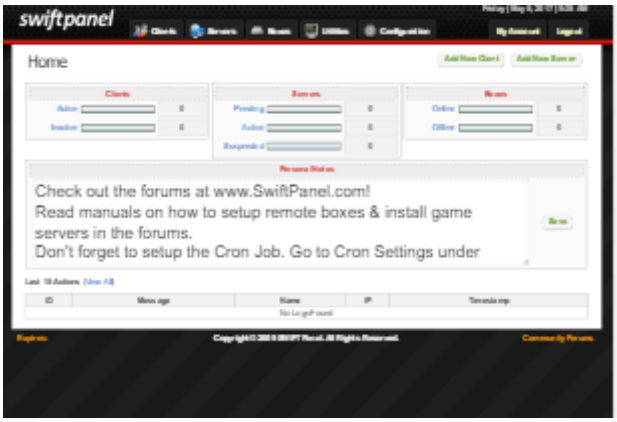
Lisas nr. 1 on välja toodud ka teisi mobiilivaateid. Rakendus sai alguse kui vabavaralise, avatud lähtekoodiga lahendus *SwiftPanel-i* asendamiseks. *SwiftPanel* on sarnane rakendus autori loodule, kuid paljude puudustega. Kuna autor oli eelnevalt ise kasutanud *SwiftPanel-it*, leidis ta, et *SwiftPanel-i* puudused ning miinused võiksid olla parandatud ning ületatud autori loodud rakenduses. Tabelis 1 on võrdlus autoriloodud rakendusele ning *SwiftPanel-ile*.

Tabel 1. *SwiftPanel*-i ja *Q3Panel*-i võrdlus

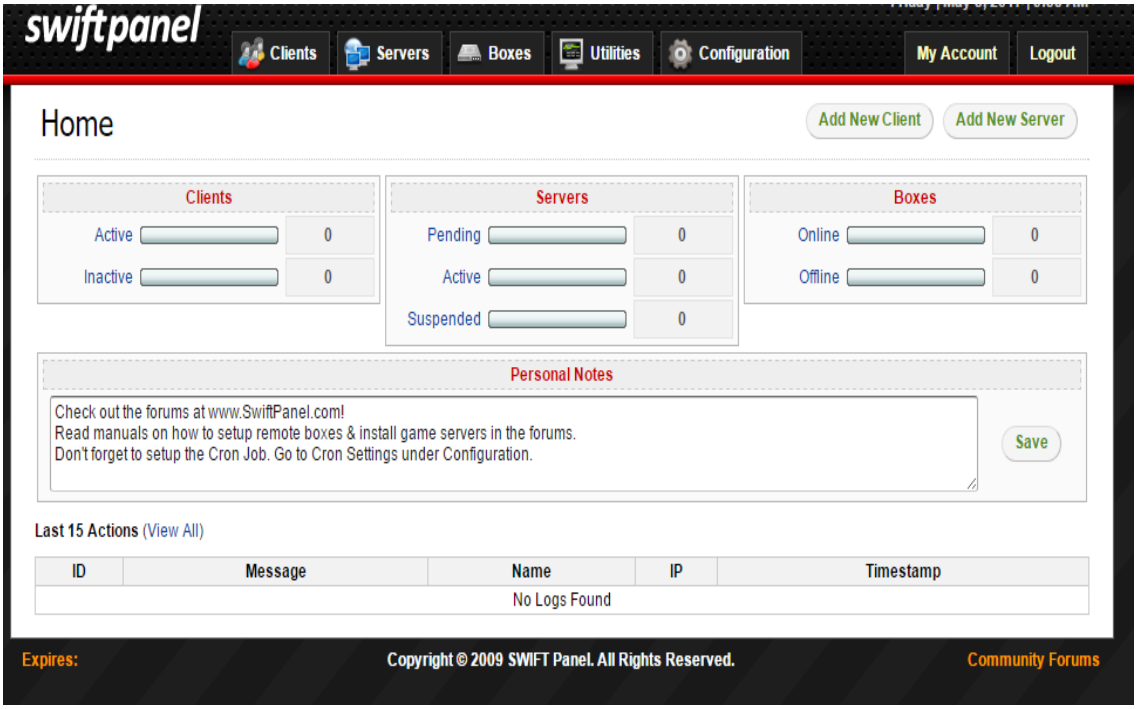
Funktsionaalsus	***Swiftpanel***	***Q3Panel***
Lihtne installatsioon (kõik veebipõhine)	Ei (konfiguratsioonifaili peab esmalt ise ära seadistama)	Jah
Mitu:Mitu suhted serverite ja kasutajate vahel	Ei (ühel kliendil võib olla mitu serverit, aga üks server ei saa olla mitmel kliendil)	Jah
Veebipõhine FTP liides	Jah (väljaarvatud failide allalaadimine)	Jah
RWD (telefonivaade, arvutivaade)	Ei	Jah
Avatud lähtekoodiga	Ei (krüpteeritud *PHP ionCube*-ga [42])	Jah

Tasuta	Ei	Jah
CRON [26] (serverite automaatne kontroll)	Jah, aga mittetoimiv (ei suuda kontrollida servereid)	Jah (kontrollib nii majutusserverit kui ka mänguserverit)
Kasutajatoe piletite süsteem	Ei, aga oli planeeritud	Jah
Ühtne liides kõigil kasutajatel	Ei, administraatoritel ja kasutajatel erinev liides ning erinev URL.	Jah
E-mail teenus	Ei	Jah
Quake 3 põhistele mängudele suunatud	Jah	Jah
Erinevad logide astmed	Ei (ainult põrunud sisselogimised)	Jah
Muudetav kujundus	Ei (ka värviskeeme pole võimalik muuta)	Jah
Serverite keelamine	Ei (kasutaja saab lukustada)	Jah
Erinevad kasutajatasemed	Kasutaja ja administraator	Keelatud kasutaja, tavakasutaja, serveri omanik ning rakenduse administraator
Mitmed rakenduseadministraatorid	Ei (on ainult üks administraatori kasutajakonto)	Jah
Veebipõhine kaugjuhtimiskonsool	Ei	Jah
Unustatud parooli funktsionaalsus	Ei	Jah
Välise andmebaasiga ühendus	Ei	Jah
Serverikonto parooli vahetus	Ei	Jah
Porditavus	Jah	Jah
AJAX kasutus	Ei	Jah
API	Ei	Jah
Laiendatavus, edasiarendatavus	Ei	Jah

Joonisel 7 ja 8 on välja toodud *Swiftpanel-i* mobiilivaade, arvutivaade ning rakenduse üleüldine välimus.



Joonis 7. SwiftPanel mobiilivaates



Joonis 8. SwiftPanel arvutivaates

Joonisel 8 nähtav mobiilivaade näitab, et rakendus pole arendatud *RWD* printsiipe kasutades ning mobiilivaates tekib olukord, kus peab pidevalt suurendama lehekülge, et näha mis on kirjas või, et saada lehel nupule pihta.

Rakendusele on arendatud ka *CRON* käsufail. *CRON* käsufail tegeleb serverite seirega. *CRON* käsu ülesseadmine teatud aja tagant jääb lõppkasutaja poolt ülesse seada. Hetkel soovitus väiksema serverite mahu korral (alla 15 serveri), on lasta *CRON* failil töödata kord minutis, kuna *CRON* pärib igalt serverikontolt algul, kas screen käsklus töötab ning kas server vastab ka käsklustele. Põrumisel kontrollitakse

kõike uuesti. Kui siis ka põrub, teavitatakse serveri kasutajaid ning server taas käivitatakse. Koodinäites nr. 1 on serveri seire välja toodud.

```
$out = $this->sendCommand($getScreenPID, true);
if (strlen(trim($out['stderr'])) === 0) {
    $pid = intval(trim($out['stdio']));
    if ($pid > 0) {
        //means screen is up, time to check the server itself.
        $q3command = Constants::$SERVER_ACTIONS['GET_Q3_SERVER_INFO'];
        $out = $this->sendQ3Command($q3command, true);
        $outArr = explode("\\", $out);
        $serverStatusGot = false;
        $currentPlayers = 0;

        for ($i = 0; $i < sizeof($outArr); $i++) {
            if (trim($outArr[$i]) === "clients") {
                $i++;
                $currentPlayers = intval(trim($outArr[$i]));
                $serverStatusGot = true;
            }
        }
        if ($serverStatusGot) {
            return array("players" => $currentPlayers);
        } else {
```

Koodinäide 1. Serveri seire

Kuna arendaja leidis, et rakendus võiks olla ka integreeritav teiste rakendustega, ehitati rakendusele ka *API*. *API* meetodid on välja toodud tabelis 2.

Tabel 2. *API*

Operatsioon	***Meetod***	***Resurss***
Sisselogimine (vajalik iga operatsiooni väljakutsel)	POST	/index.php [username], [password]
Serverite nimekirja saamine	POST	/index.php [getServers]
Spetsiifilise serveri informatsiooni saamine	POST	/index.php [getServers=serverId]
Serveri peatamine	POST	/index.php [stopServer=serverId]
Serveri käivitamine	POST	/index.php [startServer=serverId]
Serveri keelamine	POST	/index.php [disableServer=serverId]
Serveri lubamine	POST	/index.php [enableServer=serverId]

API loogikat juhib *API* klass, funktsioonide väljakutseid juhib /api/index.php fail. Hetkel leidis autor, et *API*-le pole hetkel vaja rohkem meetodeid. *API* kasutuseks peab autenditav kasutaja olema rakenduse administraatori grupis.

7 Probleemid

Peamine probleem kogu rakenduse ehitamisel oli rakenduse ja serverivaheline ühendus, käskluste jagamine ning käsklused. Autor hakkas töötama välja lahendust, kuidas jooksutada servereid kasutades PHP-d ja SSH-d nii, et rakendusele ei tekiks järgnev sõltuvus serveripoolelt. Rakendamisega tekkinud probleemid olid siis serveri käivitamine nii, et *PHP* skript ise taustal tööle ei jääks, serveri käivitamine nii, et vajaduse korral saaks kuidagi ka protsessile ligi ning ka käivitamine nii, et protsessi ID oleks võimalikult lihtsalt kättesaadav. Peale mõningaid katsetusi leidis töö autor, et parim valik oleks *Linuxi* käsklus *screen. Screen* on terminali multiplekser, millega saab tekitada mitu virtuaalset terminalisessiooni [27] . *Screeni* eelis on see, et *screenile* saab anda ka nime, mispuhul autor otsustas, et iga server saab omale eraldi serverikonto ning iga *screeni* käsk ning *screeni* protsessi nimi saab enda nimeks serverikonto nime.

Koodinäites nr. 2 on välja toodud serveri alustuskäsk.

```
"START_SERVER" => "screen -d -S {server_account} -m sh -c \"{server_startscript}\""
```

Koodinäide 2. Serveri alustuskäsk

Võti -d määrab ära selle, et *screen* hakkab taustal jooksma. -S määrab *screeni* protsessile nime, -m määrab ära, et *screen* sulgub, kui sessioon lõpeb. Kõik mis peale seda tuleb, on käsklus, mis antakse serverile. sh kutsub välja *Linux-i Bourne shell-i. Bourne shell* on käsuviiba interpreteerija [28]. sh-le edastatakse -c parameetriga serveri alustuskäsk. Peale seda, sai serverid toimima. Järgnevaks mureks oli tekitada üldse kontod majutusserverisse, lisada paroolivahetusfunktsionaalsus ning mänguserveri andmete kopeerimine, kuid see probleem oli palju väiksem. Koodinäites nr. 3 on välja toodud erinevad käsud uue serveri loomiseks koos serverikontoga.

```
"ADD_USER" => "useradd -m {server_account}"
"CHANGE_PASSWORD" => "echo \"{server_account}:{server_password}\" | chpasswd"
"COPY_GAME_FILES" => "cp -R {game_location}/* /home/{server_account}/"
"CHOWN_GAME_FILES" => "chown -R {server_account} /home/{server_account}"
```

Koodinäide 3. Serveri loomiskäsud

Esimene käsklus lisab kasutajakonto majutusserverisse. Linux platvormil on kontode lisamiseks mitmeid võimalusi, kuid töö autor soovis kasutada meetodit, mis ei jätaks *PHP* skripti tööle ja ei ootaks rohkem sisendeid kui ainult esmast käsku. *Useradd* lisab kasutajakonto ilma lisainfo pärimiseta. *Useradd* on funktsioon uue kasutaja lisamiseks [29]. *Useradd-i* -m parameeter määrab ära, et tehtavale kontole loodaks ka ta oma kodukataloog (tuues näiteks, *useradd* -m server lisaks kausta nimega server kausta */home/).* Sellele järgnevalt saab saata paroolivahetuse käsu. Kuna *useradd-iga* parooli ei seata, kasutas autor teist käsklust parooli vahetuseks. Funktsioon selle jaoks on *chpasswd. Chpasswd* on paroolide uuendamiseks batch-viisil, mistõttu saab *chpasswd-ile* saata mitmeid kasutajanime ning parooli paare, mida siis see ära vahetab. Kuna autor ei soovinud tekitada blokeerunud olukorda *PHP* skripti töölejätmisega, kus shell ootab veel sisendit, kasutas autor sellega kaasnevalt *echo* funktsionaalsust, et sööta kasutajakonto ning parool chpasswd funktsioonile sisendiks. *Echo* on funktsioon rea teksti kuvamiseks [30]. | märk ehk pipe saadab vasakpoolse käskluse väljundi parempoolse käskluse sisendiks [31]. Järgnevalt oli vaja kopeerida mängu failid serverikontole, kuna igal serveril siiski peab olema omakorda privaatne server. Seetõttu, kopeeritakse kõik mängu failid serverikontole käsklusega *cp. Cp* on funktsioon failide kopeerimiseks, -R tähistab rekursiivset kopeerimist, ehk kopeeritakse ka kaustad koos omakorda nende kaustade sisudega. Viimaseks käsuks on õiguste andmine serverikontole. See teostatakse *chown* käsuga. *Chown* on funktsioon faili omaniku/grupi vahetamiseks [32], -R määrab ära, et käsklus tuleb rakendada rekursiivselt, ehk ka erinevad kaustad ning nende sisudele tuleb rakendada käsklus. Rakendus annab kõikidele failidele ning kaustadele omanikuks serverikonto, et serverikontol oleks võimalus muuta faile.

Koodinäites nr. 4 on toodud välja serveri *screen-i* protsessi ID kättesaamise kood, serveri peatamiskood ning serveri (kaasnevalt ka serverikonto) kustutamise kood.

```
"GET_SCREEN_PID" => "screen -ls "
. "| grep -o '[0-9]\{1,5\}.{server_account}' "
. "| grep -o '[0-9]\{1,5\}' | head -1"
, "STOP_SERVER" => "kill {screen_pid}"
, "DELETE_ACCOUNT" => "userdel -fr {server_account}"
```

Koodinäide 4. Serveri protsessi ID, peatamine ning kustutamine

Esmaseks käsuks on screeni protsessi ID saamiseks. Screen -ls kuvab kõik hetkel jooksvad screeni protsessid. Pipe käsuga saadame *screen-i* väljundi grep-i sisendiks. Grep väljastab read, mis vastavad funktsiooni etteantud mustrile [35], -o parameeter määrab selle, et tagastataks ainult sobiv osa. Siinseks mustriks anname ette, et grep leiaks meile kõik vastavad read, milles on numbrid 0-st 9-ni, 1 kuni 5 korda, kuna *Linux-i* protsessi ID-si on üldjuhul 32768 [36], kui see ei ole teisiti seadistatud, ning peale numbreid peab samuti olema serverikonto nimi (mille varasemaselt screen käsuga sisestasime). Serverikonto nimi on vajalik, kuna muidu leiab grep ülesse ka kuupäevad. Pärast saadame jälle väljundi uude grep-i, kus me juba leiame ainult protsessi ID. Kuna võib tekkida olukordi, kus serveril näiteks info saatmise probleemide tõttu või mitme käsu saatmise tõttu saab serverit mitu korda alustada, siis sealt me võtame head käsuga ainult ühe rea välja. Järgnevaks käsuks on kill. Kill on käsklus, mis sulgeb protsessi [33]. Kill sisendparameetriks anname eelneva protsessi ID. Viimaseks käsuks on *userdel. Userdel* kustutab kasutajakonto [34], -f tähendab, et käsklus viiakse läbi igal juhul, isegi siis kui kasutaja on jätkuvalt sisse logitud. -r määrab ära, et ka kasutaja kodukataloog eemaldatakse. Nii kustutatakse ka serverifailid ära koos serverikontoga.

Serveri käivituskäsk: Koodinäites nr. 5 on käsk on *Quake 3* mängumootoril jooksva mängu *Soldier of Fortune* 2 serveri programmi käivituskäsk, kuid see kood toimib igal *Quake 3* mängul, kui vastavad parameetrid ära muuta. Järgnev käsk on *Quake 3* mängumootoril jooksva mängu *Soldier of Fortune 2* serveri käivituskäsk, kuid enamik sellest toimib igal *Quake 3* mängul.

```
ln -s /usr/local/games/base ~/base;
~/sof2ded +set dedicated 2 +set vm_game 0 +set fs_game 1fx +set net_port {server_port}
 +set fs_homepath /home/{server_account} +set g_clientMod "RPM"
 +set sv_maxclients {max_clients} +exec Config.cfg +set rconpassword {rconpassword}
```

Koodinäide 5. Serveri programmi käivituskäsk

Kõige esimene käsklus on kõige tähtsam. Esmaselt kopeeriti kõik serverifailid serverikontole. Aga kuna serveri otseseks jooksutamiseks ei ole vaja igakord kopeerida faile, mis ei muutu, siis rakendus kopeerib ainult failid, mis on talle ette antud. Ehk siinne lähenemine saab olla järgnev: seadistaja seab eraldi kausta, kus on olemas serveri failid, mida ei saa jagada, tuues näiteks konfiguratsioonifailid, programm ise. Ülejäänud failid saab pärida kasutajale käsuga ln. Ln on käsk

faili/kausta linkimiseks, -s parameeter määrab, et link on sümboolne. Sümboolseid linke saab kasutada ka kaustade jaoks. Tänu sellele, lingime me vajalikud staatilised failid hetkelise serverikonto kodukataloogi, et staatilisi faile ei peaks kopeerima üle ning iga serveriga kaotada gigabait mälumahtu. Ülejäänud käsklus on serveri töölepanemiskäsklus.

Käsklustes kõik { } märkide vahel olevad märksõnad vahetatakse rakenduse poolt välja nende tõeliste parameetrite vastu.

8 Kasutajate tagasiside

Bakalaureusetöö kirjutamise hetkel on rakenduses peale autori veel seitse kasutajat, kellest neli on aktiivsed kasutajad. Autor andis ligipääsu neljale kasutajale varajases etapis, et oleks näha, kuidas rakendus areneb ning kas neil tekib küsimusi või soovitusi. Pärast arendust palus autor neil neljal kasutajal kirjutada lühikese ülevaate, mida arvavad nemad rakendusest, mis on nende arvates puudu, kas nad leidsid vigu ja üleüldiseid kommentaare.

Kasutajad võtsid võrdluseks eelnevalt väljatoodud *SwiftPanel-i*, mis on sarnane rakendus autori loodud rakendusele. Oluliseks peeti, et kasutajad said uue rakenduse arendamisest teada suhteliselt alguses ja neil oli võimalus autorile ettepanekuid teha. Heaks näitajaks oli see, et rakenduse arendaja pidevalt uuris, mida võiks lisada ja kas on veel vigu mida parandada.

Välja toodi, et serverite vaates on võib-olla hetkel palju informatsiooni, nagu näiteks serveri *FTP* konto ja parool. Väideti, et serveri detailsemas vaates tuleb kogu see informatsioon välja ning leiti, et võib-olla pole seda informatsiooni otseselt serveri vaatele vaja. Töö autor pidas ebavajalikuks seda muuta, kuna nii säilib üldine hea ülevaade neile kes ei taha detailvaatesse minna. Serveri administraatori rollis on võimalik ka serverite konfiguratsiooni muuta, mis tavakasutajatel *SwiftPanel-is* polnud võimalik (selleks pidi olema rakenduse administraator).

Suurimaks õnnestumiseks pidasid kasutajad erinevad käskluste edastamist serverihalduse lehel. Kuigi SwiftPanel-il oli olemas ka veebipõhine *FTP* liides, on *Q3Panel-i* veebipõhine *FTP* liides kiirem, lihtsam kasutada ning laseb teha kiireid käsklusi mida SwiftPanel-iga ei saanud. Seda just failide ja kaustade kustutamisel ning ümbernimetamisel. Sellel saab muuta faili sisu ilma lehelt lahkumata. Kasutaja eeldas, et veebipõhist *FTP* liidest pole võimalik lisada rakendusse sellisel kujul, kuna *FTP* liides, mis oli *SwiftPanel*-is, oli ainult tekstifailide muutmiseks. Kõik kasutajad ütlesid oma tagasisides, et uus rakendus on lihtsam ja kasutajasõbralikum, ei pea kõike muudatusi läbi *FTP* rakenduse tegema.

Veel tuuakse välja kasutajatoe piletite süsteemi, mis loodud rakenduses on hea, kuna selles saab jätta probleemid rakendusse koos logiga, kui peaks juhtuma, et serveriga on probleeme. Lihtsalt kasutatavat kasutajatoe piletite süsteemi polnud *SwiftPanel-is* üldse olemas. On väga lihtne registreerida uusi kasutajatoe pileteid, lisada

kommentaare või lihtsalt lugeda kõiki pileteid, mis on kasutajale edastatud. Kasutajad saavad teavituse, kui piletit on uuendatud ning detailne logimine on olemas igal erineval sõnumil piletis.

Kasutajate vaade sarnaneb *SwiftPanel-i* kasutajate vaatega, aga see on detailsem ning kasutajatel on rohkem gruppe. Suureks eeliseks nüüdsel rakendusel on see, et enam ei pea olema mitu erinevat kasutajakontot erinevale serverile ning kõiki servereid saab hallata ühe kasutajakonto alt. Samuti on ka logid nähtaval, mis olid osaliselt ka olemas *SwiftPanel-is,* kuid autori poolt loodud rakenduses on logid palju detailsemad ning informatiivsemad. Kasutajad tõid välja mõned vead rakenduses, nimelt pääses rakenduse logidele ligi. Kuigi logisi ta ei näinud, oli vaade tal linkide nimekirjas olemas. See tekitas veidi segadust ning autor parandas vea.

Teiseks veaks oli serveri info logimine. Hetkel oli serveri info logimine viidud rakenduse logidesse, kuid seal oli nähtaval ainult serveri identifikaatorväli ning serveri kohta informatsiooni peale selle ei olnud. Autor parandas ka selle vea ning lisas logimise serveri logidesse.

Rakenduse esmase versiooni väljalaskmisel tõi üks kasutaja välja kaks viga. Üheks oli see, et rakendusest välja logimisel jäi kasutaja lõksu sisselogimise-väljalogimise ringi, kus iga kord kui ta üritas sisse logida, logis rakendus ta automaatselt ka välja. Teiseks tõi kasutaja välja selle, et veebipõhise *FTP* liideses failinime muutmine viis peale muudatust kasutaja tagasi juurkataloogi. Autor parandas need kaks viga pärast tagasiside saamist ära.

Välja on toodud, et rakendusel puudub serveri väljundlogi, ehk mis hetkel serveris toimub. Serveri väljundlogi koosneb kõigest, mis serveris hetkel toimub. Kuid kuna autor kasutas *screen-i* serverite jooksutamiseks, tekitaks see olukorra, kus rakendus peab pärima iga kord akna avamist ning sulgemist lihtsalt andmete saamiseks, mis tekitaks lihtsalt serverile palju koormust, ning seetõttu otsustas, et seda funktsionaalsust pole otstarbekas rakendusse lisada.

Sisseehitatud kaugjuhtimiskonsooliga saab saata käsklusi serverile ning rakendus tagastab serveri vastuse kasutajale. Kasutajate sõnul töötab see väga hästi ja hoiab palju aega kokku. Veebipõhise kaugjuhtimiskonsooli kohta tõid kasutajad välja selle, et kuigi sellel ei ole juhendit, oskavad teadlikumad kasutajad seda kasutada. Autori väide siinjuures on see, et *Quake 3* mootoril jookseb palju mänge, ning igal mängul

võib olla oma meetod ning oma funktsionaalsused, mistõttu ei leidnud autor, et oleks vajadust defineerida erinevad käsud juhendiga veebipõhisesse kaugjuhtimiskonsooli.

Kasutajad leidsid, et majutusserveri halduse leht puhas ja selles saab uue majutusserveri lihtsalt üles seada. Informatsiooniveerg paremal on hea lisa. See näitab informatsiooni rakenduse kohta, nagu näiteks mitu serverit ja mitu kasutajat on hetkel rakenduses olemas. See annab samuti ka ülevaate kuidas on rakendus ülesse seatud, nagu näiteks meilindus ning autentimine. Pärast esmaste väljade täitmist uuele mänguserverile, nagu näiteks majutusserver, maksimaalne mängijate arv ning *FTP* konto, saab sa kohe aru, et horisontaalset ala *Q3Panel-is* on hästi ära kasutatud. On näha suur tabel kogu asjakohase informatsiooniga.

Üks kasutajatest on olnud seotud rakenduse arendusega lähemalt, kui teised kasutajad. Pärast rakenduse arenduse lõppu, ehitas kasutaja ka laienduse autori rakendusele. Kuna autor majutab selle kasutaja mänguserverit, siis soovis ta lisada rakendusele ühe lisa, mis automaatselt uuendaks serveril olevat tarkvara iga teatud aja tagant siis, kui uus tarkvara on saadaval ning serveris hetkel mängijaid pole. Kasutaja leidis, et tänu objektorienteeritud lähenemisele ning funktsionaalsuste kommenteerimisele, oli selle laienduse ehitus vägagi lihtne ning mitte aega nõudev. Autor võimaldas kasutajale ühe andmebaasi tabeli ning kasutaja arendas ise laienduse välja, mis hetkel ka autoripoolses töötavas rakenduses toimib edukalt.

Kasutajad leidsid, et uus rakendus näeb parem välja nii telefonis kui ka arvutis ning samuti väideti, et huvitav lisa on värviprofiilide muutmine. Serverite haldus on lihtne ja isegi mitte nii arvutiteadlik kasutaja saab hakkama rakenduses. Kuna varem oldi suur *SwiftPanel-i* kasutaja, väideti, et vajatakse veidi harjumisaega, kuna rakenduse struktuur on erinev *SwiftPanel-ist.* Tasuta kasutatavas ja avatud lähtekoodiga rakenduses on saanud kõik asjad korda mida *SwiftPanel-is* ei saanud kunagi. Selle paigaldus on lihtne, samuti ka navigeerimine ning kasutajad ütlesid, et kõik on perfektne. Juba mõnda aega oli otsitud parimat serveripaneeli, et majutada oma *Quake 3* mootoril põhinevaid mänguservereid, ja pärast kõiki neid aastaid on erakordselt hea rakendus olemas. Ühe kasutaja tagasiside originaalvariant on vaadeldav lisas nr. 4.

9 Kokkuvõte

Bakalaureusetöö põhieesmärk oli luua rakendus, mis haldaks *Quake 3* mootoril jooksvaid mänguservereid ning lihtsustada kogu serverihaldusprotsessi. Enne töö algust tehtud analüüs, mida rakenduses on vaja, aitas aru ka saada autoril, kui palju veel on arendustegevusega vaja tegeleda.

Töö tulemuseks on reaalne ning kasutatav rakendus. Lõplikus rakenduses sai kõik püstitatud eesmärgid täidetud ning vajalik funktsionaalsus loodud. Samuti sai lisatud kasutajate tagasisidel põhinevaid erinevaid funktsionaalsusi, mida nemad algselt soovisid näha. Kuigi rakendus on valmis ning hetkel ei ole plaane midagi juurde arendada, muutub see eeldatavasti kasutuse käigus. Tänu rakenduse lähtekoodi avatusele, on rakendust võimalik ka teistel edasi arendada ning tänu funktsionaalsuse kommentaaridele on rakenduse edasiarendamine ka lihtsam. Töö lugeja peaks saama siit ülevaate rakenduse ülesehitusest ning osade probleemide lahendamisest.

Kasutatud kirjandus

[1] „What is MySQL?“, [WWW] URL https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html (05.05.2017)

[2] „What is PHP?“, [WWW] URL http://php.net/manual/en/intro-whatis.php (05.05.2017)

[3] „Responsive web design“, [WWW] URL https://en.wikipedia.org/wiki/Responsive_web_design (05.05.2017)

[4] „Welcome to Netbeans“, [WWW] URL https://netbeans.org/ (05.05.2017)

[5] „jQuery“, [WWW] URL https://jquery.com/ (05.05.2017)

[6] „Bootstrap – The world’s most popular mobile-first and responsive front-end framework“, [WWW] URL http://getbootstrap.com/ (06.05.2017)

[7] „Angle – Bootstrap Admin Template“, [WWW] URL https://angle-on-rails.herokuapp.com/dashboard/dashboard_v1 (06.05.2017)

[8] „HTML“, [WWW] URL https://en.wikipedia.org/wiki/HTML (07.05.2017)

[9] „DataTables | Table plug-in for jQuery“, [WWW] URL https://datatables.net/ (07.05.2017)

[10] „Select2 – The jQuery replacement for select boxes“, [WWW] URL https://select2.github.io/ (08.05.2017)

[11] „Toastr by CodeSeven“, [WWW] URL https://codeseven.github.io/toastr/ (10.05.2017)

[12] „MySQL“, [WWW] URL https://en.wikipedia.org/wiki/MySQL (10.05.2017)

[13] „PHP: password_hash – Manual“, [WWW] URL http://php.net/manual/en/function.password-hash.php (05.05.2017)

[14] „JQuery“, [WWW] URL https://en.wikipedia.org/wiki/JQuery (05.05.2017)

[15] „Cascading Style Sheets“, [WWW] URL https://en.wikipedia.org/wiki/Cascading_Style_Sheets (07.07.2017)

[16] „JavaScript“, [WWW] URL https://en.wikipedia.org/wiki/JavaScript (10.05.2017)

[17] „GitHub“, [WWW] URL https://github.com/ (05.05.2017)

[18] „PHP: require_once – Manual“, [WWW] URL http://php.net/manual/en/function.require-once.php (10.05.2017)

[19] „POST (HTTP)“, [WWW] URL https://en.wikipedia.org/wiki/POST_(HTTP) (10.05.2017)

[20] „Hypertext Transfer Protocol“, [WWW] URL https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol (10.05.2017)

[21] „Server commands howto“, [WWW] URL ftp://ftp.idsoftware.com/idstuff/quake3/docs/server.txt (05.05.2017)

[22] „Whoami“, [WWW] URL https://en.wikipedia.org/wiki/Whoami (10.05.2017)

[23] „Secure Shell“, [WWW] URL https://en.wikipedia.org/wiki/Secure_Shell (09.05.2017)

[24] „Ajax (programming)“, [WWW] URL https://en.wikipedia.org/wiki/Ajax_(programming) (05.05.2017)

[25] „Bootstrap Modals“, [WWW] URL https://www.w3schools.com/bootstrap/bootstrap_modal.asp (05.05.2017)

[26] „Cron“, [WWW] URL https://en.wikipedia.org/wiki/Cron (10.05.2017)

[27] „Screen User’s manual“, [WWW] URL https://www.gnu.org/software/screen/manual/screen.html#Overview (10.05.2017)

[28] „Bourne shell“, [WWW] URL https://en.wikipedia.org/wiki/Bourne_shell (10.05.2017)

[29] „useradd(8) – Linux man page“, [WWW] URL https://linux.die.net/man/8/useradd (05.05.2017)

[30] „echo(1): line of text – Linux man page“, [WWW] URL https://linux.die.net/man/1/echo (05.05.2017)

[31] „Linux Tutorial – 11. Learn Piping and Redirection“, [WWW] URL http://ryanstutorials.net/linuxtutorial/piping.php (05.05.2017)

[32] „Chown“, [WWW] URL https://en.wikipedia.org/wiki/Chown (06.05.2017)

[33] „kill(1): terminate process – Linux man page“, [WWW] URL https://linux.die.net/man/1/kill (06.05.2017)

[34] „userdel(8): delete user account/related files – Linux man page“, [WWW] URL https://linux.die.net/man/8/userdel (06.05.2017)

[35] „GNU Grep 3.0“, [WWW] URL https://www.gnu.org/savannah-checkouts/gnu/grep/manual/grep.html (06.05.2017)

[36] „c – Maximum PID in Linux – Stack Overflow“, [WWW] URL http://stackoverflow.com/a/6294196 (10.05.2017)

[37] „The classic email sending library for PHP“, [WWW] URL https://github.com/PHPMailer/PHPMailer (05.05.2017)

[38] „Marketing & Transactional Email Service | SendGrid“, [WWW] URL https://sendgrid.com/ (05.05.2017)

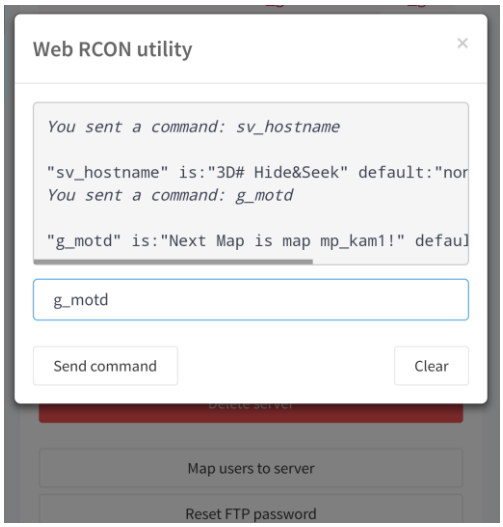
[39] „SendGrid User Guide – SendGrid Documentation“, [WWW] URL https://sendgrid.com/docs/User_Guide/index.html (05.05.2017)

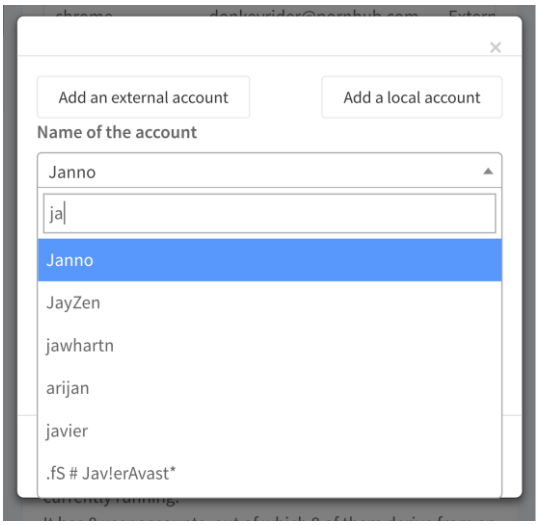
[40] „PHPMailer“, [WWW] URL https://en.wikipedia.org/wiki/PHPMailer (05.05.2017)

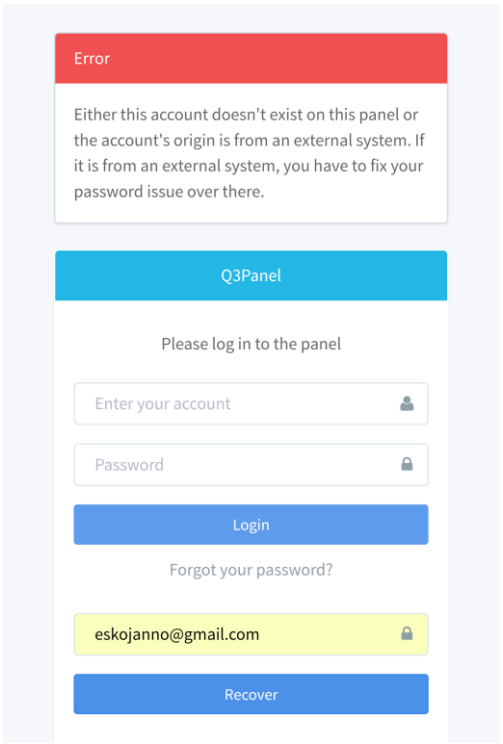
[41] „Libscore“, [WWW] URL http://libscore.com/#libs (12.05.2017)

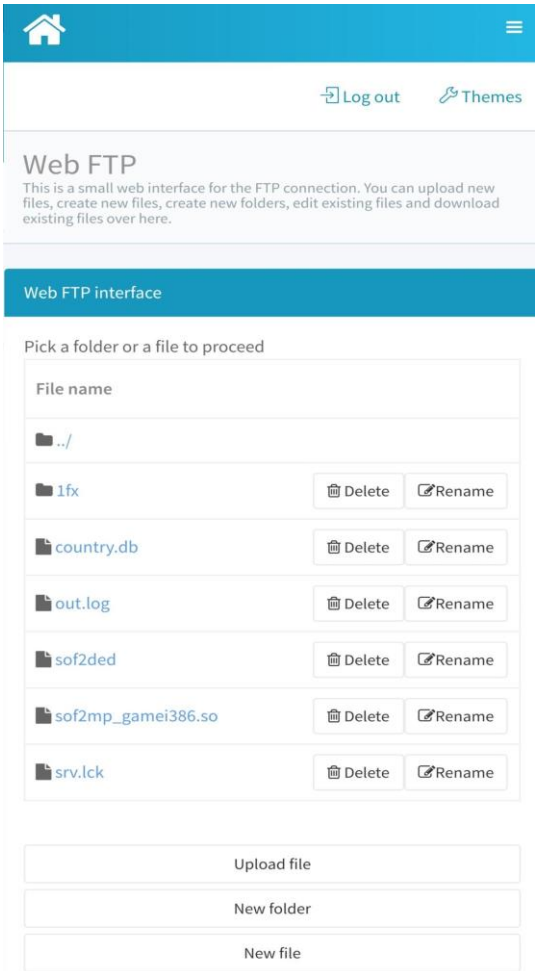
[42] „ionCube“, [WWW] URL https://en.wikipedia.org/wiki/IonCube (21.05.2017)

Lisa 1 – Rakenduse erinevad vaated mobiilis







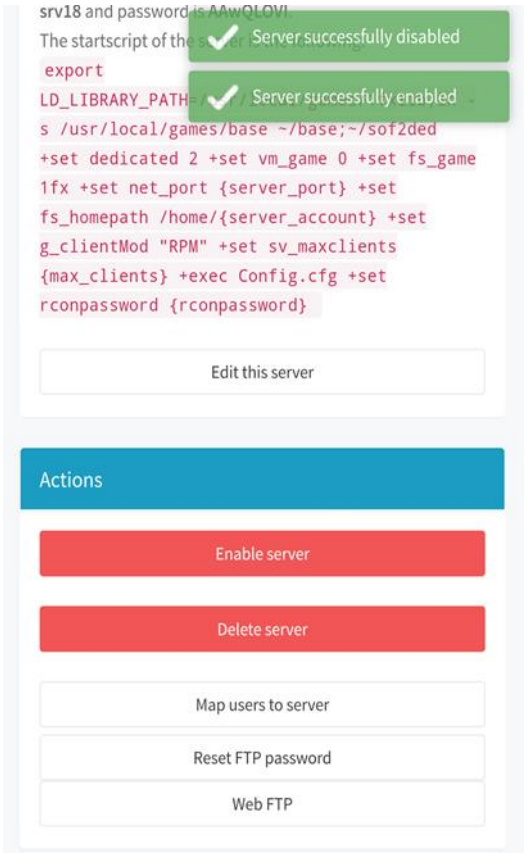


Lisa 2 – Välise kasutajakonto andmete saamise šabloon

```
/**
 * Replaces the missing parts in an external query.
 * @param string $query The initial query with the {ext_ tags.
 * @param string $ext_usrtable The external usertable.
 * @param string $ext_usrtable_id The external usertable primary key (id field)
 * @param string $ext_usrname The external usertable username field.
 * @param string $ext_psw The external usertable password field.
 * @param string $ext_email The external usertable email field.
 * @return string The query with the parts changed.
 */
static function getExternalQuery($query, $ext_usrtable, $ext_usrtable_id = null,
        $ext_usrname = null, $ext_psw = null, $ext_email = null) {
    return str_replace("{ext_usrtable}", $ext_usrtable,
        str_replace("{ext_usrtable_id}", $ext_usrtable_id,
            str_replace("{ext_usrname}", $ext_usrname,
                str_replace("{ext_psw}", $ext_psw,
                        str_replace("{ext_email}", $ext_email, $query)
                )
            )
        )
    );

"GET_EXT_USER_BY_ID" => "SELECT {ext_usrtable_id}, {ext_usrname}, "
"{ext_email} FROM {ext_usrtable} WHERE {ext_usrtable_id} = ?"
```

Lisa 3 – Toastr




```
/**
 * Enables a server.
 * @param {int} server_id The server ID.
 * @returns {void} Returns nothing.
 */
function enableServer(server_id) {
    $.post(".", {
        enableServer: 1,
        server_id: server_id
    }, function(data) {
        data = JSON.parse(data);
        if (typeof data.error !== "undefined") {
            toastr.error(data.error);
        } else if (typeof data.msg !== "undefined") {
            $("#stopServer").hide(500);
            $("#startServer").show(500);
            $("#enableServerBtn").hide(500);
            $("#disableServerBtn").show(500);

            toastr.success(data.msg);
        }
    });
}
```

Lisa 4 - Kasutaja Ane-Jouke Schat tagasiside

Review written on May 10th, 2017.

Introduction

Over the years I've hosted lots of games, all of which are based on the Quake 3 engine. A game server panel I've been particularly fond of has always been SwiftPanel, mainly due to it’s easy-to-use interface and good support for Quake 3 engine-based games. I also believe a good panel should cost no money, and source code should be freely available. When I heard Janno made something similar to SwiftPanel, but made improvements on all fronts, I couldn't wait to try it out. Thus, the testing of q3panel began.

This review is based on my previous experience with SwiftPanel, and can therefore be considered a direct comparison between the two.

Installation

The installation of q3panel is easy. It requires some basic web server knowledge to get going (install Apache, PHP 7, some additional modules), but overall the experience is smooth and the panel is up within no time. Another big bonus that, if you miss modules installed, it will actually hint what packages you might need on Debian based distributions. Those are even a good indicator on non-Debian based distributions. A good, easy to follow installation procedure.

This is unlike SwiftPanel, where installation is a nightmare due to encoder used on the source code, ionCube. Getting the appropriate decoder up and running for the PHP version installed can take some time, where you’ll get zero feedback from SwiftPanel: all you see is an empty white page with no details of what went wrong. After you finally load the correct decoder, installation is easy to follow, just like q3panel.

One thing I would've liked is support for more database types, next to MySQL. The addition of PostgreSQL or SQLite would've been nice. Using SQLite would mean no fully fledged SQL server would have to be setup on the host. Still, *SwiftPanel* doesn't support anything other MySQL either, so no harm done there.

A clear win for q3panel.

First impressions

The q3panel looks clean, is responsive, uses modern technologies and gives a good first impression. Little things such as theming are subtlety introduced and don’t get in the way of getting things done.

Getting your away around the panel is easy, and there are small hints everywhere making the first usage a pleasant one. So after getting acquainted with the new interface, I went on to configure some users.

User management

User management is another clear win for the q3panel. *SwiftPanel* only supports basic users. Using q3panel, you can connect the panel to an existing database, making it easily integrated in other applications already hosted, also not requiring different passwords.

What I also like is the fact that you can connect users to multiple servers in q3panel. In SwiftPanel, each server had one user, making sharing user accounts inevitable.

What I would like to see in q3panel, even though out of scope for the initial project, is the customization of user groups. Right now you have three groups, and even though divided logically, are still pretty static. Customizing groups would mean new roles are easily created yourself, such as e.g. moderator of specific tasks. This would allow for more flexibility.

Host management

Not really a lot to say about the host management. Like everything, the page is neat and you can easily set a host up.

The information column on the right is a nice touch. It shows information regarding the panel, like how many servers and users you have in total. It also gives a quick glimpse on how things are setup, like mail and authentication.

SwiftPanel didn't have anything fancy, but the page still got the job done, and quite easily so.

G*ame server management*

After you fill out the basic things for the new server, such as host, max players and FTP account, you immediately notice how well the available horizontal space is used

in q3panel when you're guided back to the overview. You are greeted with a massive table, showing you all relevant information, without being overwhelming. Job well done.

Managing a server is where things get interesting. The in-depth server info doesn't show a table with relevant info, but tells you the details in a story mode. I was initially a bit skeptical about this, but after getting used to it, it does allow you to view the information in an efficient way.

The actions on the server management page is where q3panel really shines. Even though *SwiftPanel* also had a Web FTP, this Web FTP faster, is easier to overlook and allows you to do quick actions you couldn't do with SwiftPanel, such as deleting and renaming files and directories. You can even modify file contents without ever being redirected to a different page. Clear win.

Another thing I liked was the in-built RCON (remote console) support. With this, you can send commands to the server and get the output of it. It works really well and actually saves you quite some time. Where I used to head back in the game to check something out, I can now do it all from the web RCON utility. Awesome! Maybe in the future also support to read the console for a period of time instead of sending just one command and retrieving its output? ;-)

Ticket management

Easy to use tickets between users is something *SwiftPanel* didn't have at all, but q3panel has! It’s easy to register new tickets, add commentary or just view all issues assigned to you. Users are notified when a ticket is updated and a detailed logging is kept for each individual message in the ticket.

Conclusion

Coming from SwiftPanel, using q3panel is exciting to say the least. Things that *SwiftPanel* never got right are finally dealt with, in a free to use and open source panel. Its installation is easy, as is navigation and just about everything really. I've been looking for the best server panel to host my Quake 3-based games on for quite some time now, and after all those years I've finally found it. Well done Janno, you put together an extraordinary panel!