TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Tarkvarateaduse instituut

Sten-Hendrik Pihlak 134672IAPB

VEEBIRAKENDUS MUUSIKALISE KUULMISE ARENDAMISEKS

Bakalaureusetöö

Juhendaja: Jaagup Irve
Tehnikateaduste magister

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: Sten-Hendrik Pihlak

22.05.2017

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua veebirakendus muusikalise kuulmise treenimiseks. Ülesehitatud veebirakenduse sihtgrupp on isikud, eriti kitarristid, kes soovivad arendada enda muusikalist kuulmist.

Lõputöö käigus luuakse veebirakendus, milles on kasutajal võimalik valida kolme erineva mänguvormi vahel ning harjutada erinevate raskustasemetega. Valminud veebirakendust on võimalik kasutada nii nutiseadmes kui ka lauaarvutis.

Lõputöö koosneb analüüsist, veebirakenduse arendusest, kasutatud tööriistadest ning hinnangust. Analüüsi käigus seatakse lõputööle nõuded, millele bakalaureusetöö peab vastama. Veebirakenduse arenduses implementeeritakse analüüsi käigus seatud nõuded ning arutletakse veebirakenduse arendusest ja tehnilisest poolest. Lisaks tuuakse kasututatud tööriistade all välja ka tööriistad, mille abil lõputööd realiseeriti. Kogu lõputöö võetakse kokku hinnangus, kus autor annab enda järelduse valminud lõputööst ning selle implementatsioonist.

Veebirakendus on realiseeritud Node.js platformil.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 32 leheküljel, 6 peatükki, 14 joonist.

Abstract

Web application for improving musical ear

The purpose of this thesis is to create a web application which helps to train the musical ear. Built web application is aimed towards individuals who wish to train their musical ear. It is specially oriented towards guitarists.

During the thesis a web application is made where the user has the choice to choose between three game types which can be played on different difficulty levels. The following web application can be used both in mobile devices and computers.

The thesis consists of analysis, development of the web application, used tools and evaluation. During analysis the goals of the thesis are set. In the web application development those goals are being implemented. In addition there is an argumentation regarding the web application development and technical side. Under the used tools paragraph tools which were used in order to implement the web application are listed. The thesis is finalized in the evaluation paragraph where the author provides his conclusions on the thesis and the implementation.

The web application is implemented in Node.js.

The thesis is in Estonian and contains 32 pages of text, 6 chapters, 14 figures.

Lühendite ja mõistete sõnastik

Ajax	*Asynchronus Javascript And XML* – tehnoloogia päringute tegemiseks ning andmete vastuvõtmiseks
Bootstrap	Veebiraamistik veebirakenduste disainimiseks
CDN	*Content Delivery Network* – sisu pakkumise viis
DOM	*Document Object Model* – Dokumendi objektimudel
GET	Päringumeetod andmete saamiseks
HTML	*Hypertext Markup Language* – Hüpterteksti märgistuskeel
HTTP	*Hypertext Transfer Protocol* – Hüperteksti edastusprotokoll
JavaScript	Objektorienteeritud programmeerimiskeel
jQuery	Teek *JavaScript* hõlbustamiseks
JSON	*JavaScript Object Notation* - andmevahetusvorming
MVC	*Modal-View-Controller* – Mudel-Vaade-Kontroller arhitektuurimudel
Node.js	Sisend-väljundplatvorm veebirakenduste loomiseks
POST	Päringumeetod andmete salvestamiseks, saamiseks ja muutmiseks

Sisukord

Jooniste loetelu

1 Sissejuhatus

Veebirakendus on arvutiprogramm, mis kasutab veebibrausereid ning veebitehnoloogiaid, et sooritata interneti vahendusel ülesandeid [1]. Veebirakenduses eristatakase kliendipoolset rakendust ning serveripoolset rakendust.

1.1 Eesmärk

Lõputööna valminud veebirakenduse eesmärk on pakkuda kasutajale võimalust arendada tema muusikalist kuulmist. Veebirakendus võimaldab kasutajal valida kolme erineva mänguvormi vahel:

- Üksikute nootide mänguvorm – kasutajale mängitakse ette üksikuid noote, abistavaks informatsiooniks on ette antud helistik nimetus ning positsioon.
- Akordide mänguvorm – kasutajale mängitakse ette akorde.
- Kombineeritud väljakutse – kasutajale mängitakse ette akorde ning noote. Nootide ettemängimsel on abistavaks informatsiooniks kuvatud helistiku nimetus ning positsioon.

Nootide mänguvormile on sisse ehitatud funktsionaalsus, mis vastavalt ettemängitud skaalale prioritiseerib nootide ettemängimise valikus noote, mis on lähestikku eelnevalt mängitud nootidele.

Kasutajalt küsitakse enne mängu algust hinnangut tema oskuste kohta, mille põhjal otsustatakse mängu raskustase. Kui kasutaja alahindab oma oskust, siis muudetakse vastava mänguvormi taset lihtsamaks ning kui kasutaja ülehindab oma oskust, siis muudetakse vastava mänguvormi taset keerulisemaks.

- Üksikute nootide mänguvormi puhul hakatakse ette mängima noote teistest heliridadest.

- Akordide mänguvormi puhul võetakse kausutusele keerulisemaid akorde.
- Kombineeritud väljakutse puhul hakatakse ette mängima keerulisemaid akorde ning lisaks akordidele ka noote.

Kombineeritud väljakutse ja akordide mänguvormi akordi ettemängimisel on kasutajal võimalik sisendisse kaasa anda enda akord ning selle täpsustus.

Kombineeritud väljakutse ja nootide mänguvormi noodi ettemängimisel kuvatakse kasutajale interaktiivne kitarrikael, kus kasutaja saab valida vastuse nootide peale vajutades.

2 Analüüs

Järgnevas peatükis kirjeldatakse lõputööna valminud veebirakenduse funktsionaalseid- ja mittefunktsionaalseid nõudeid.

2.1 Funktsionaalsed nõuded

1. Kasutaja saab valida mänguvormi, kus mängitakse ette ainult akorde.
2. Kasutaja saab valida mänguvormi, kus mängitakse ette ainult noote.
3. Kasutaja saab valida mänguvormi, kus mängitakse ette nii akorde kui ka noote.
4. Veebirakendusel on vasakul pool menüü mänguvormide valikuvõimalustega.
5. Kasutaja saab sisestada enda vastuse.
6. Nootide mänguvormi puhul saab kasutaja valida vastuse interaktiivse kitarrikaela pealt.
7. Kombineeritud väljakutse noodi ettemängimisel saab kasutaja valida vastuse interaktiivse kitarrikaela pealt.
8. Akordide mänguvormi puhul saab kasutaja sisestada vastuse ning valikukastist valida akordi täpsustuse.
9. Kombineeritud väljakutse akordi ettemängimisel saab kasutaja sisestada vastuse ning valikukastist valida akordi täpsustuse.
10. Kasutaja saab anda hinnangu enda tasemest.
11. Süsteem kuvab kasutajale tema hetkese punktiseisu.
12. Süsteem kuvab vastavalt kasutaja sisendile tulemi, kas vastus on vale või õige.
13. Süsteem muudab raskustaset vastavalt kasutaja tulemusele.

14. Süsteem mängib helifaili iga mänguvormi alustamisel.

15. Süsteem mängib sama helifaili uuesti kui kasutaja vastab valesti.

16. Õige vastuse puhul mängitakse kasutajale ette uus helifail.

17. Kasutajale kuvatakse väike tutvustus kasutaja valitud mängivormist.

2.2 Mittefunktsionaalsed nõuded

1. Veebirakenduse kasutajaliides on ingliskeelne.

2. Veebirakenduse kood peab olema ingliskeelne.

3 Veebirakenduse arendus

Järgnevavates alampunktides räägitakse lõputööna valminud veebirakenduse arendusest.

3.1 Veebirakenduse arenduslugu

Lõputööna valminud veebirakenduse arendamine algas programmeerimiskeele valikust – algselt oli plaan teha veebirakendus PHP programmeerimiskeeles, kuid lõppkokkuvõtteks otsustati Node.js kasuks JavaScript tõttu, mis on rakendatav nii serveri kui ka kliendipoolel. Samuti mõjutas valikut arendaja soov tutvuda lõputöö käigus tehnoloogiaga, millega lõputöö esitaja polnud varem kokku puutunud.

Veebirakendust kavatseti arendada MVC mustri põhjal ning lähtuvalt sellest valiti Node.js arenduse põhjaks minimaalne Node.js raamistik Express. Express valiti tema minimaalsuse ning lihtsuse tõttu.

Olulise osana kavandati veebirakendusse õppemudel ning õppija hindamise algoritm. Iga mänguvormi eelduseks on kasutaja algse hinnangu sisestamine, mida hakatakse vastava algoritmiga jälgima ning mis tagab eeldused järgmise tasemele jõudmiseks.

Veebirakenduse vaadete kuvamiseks oli algselt plaanis kasutada HTML malle, kuid otsustati EJS kasuks tema täiendava funktsionaalsuse poolest, mis ühendab serveri poolt saadetud andmed malliga.

Veebirakendus arendati lokaalses Windows operatsioonisüsteemiga masinas. Veebirakenduse arenduse lõpus tõsteti valminud veebirakendus üle Veebimajutuse [2] serverisse.

Arenduse põhietapid:

1. Menüü loomine.

2. Mänguvormide EJS mallide loomine ja kujunduse loomine Bootstrap põhjal.

3. Route.js faili loomine, kus kirjeldati ära esimesed GET meetodid, mille abil kuvati iga aadressi puhul õige vaade.

4. Esialgne helifailide lindistamine.

5. Akordide ja nootide mänguvormide loogika arendamine ning rakendamine POST meetodi abil.

6. Kombineeritud väljakutse jaoks akordide ning nootide loogika kombineerimine ning rakendamine POST meetodi abil.

7. Interaktiivse kitarriskaala rakendamine.

8. Veebirakenduse üleviimine kohalikust masinast serverisse.

3.2 Veebirakenduses kasutatud tehnoloogiad

Rakendus on loodud Node.js abil, mis põhineb JavaScript programmeerimiskeelel. Node.js kasutab sisend/väljund mudelit, mis garanteerib platvormile kiiruse ning kergekaalu. Lisaks on Node.js suureks eeliseks on JavaScript, sest nimetatud programmeerimiskeel on rakendatav nii kliendi- kui ka serveripoolel. Selle omaduse tõttu saame kasutada *back-end* ja *front-end* poolel samasuguseid mustreid, muutes arenduse mugavamaks. Node.js rakendatakase enamasti reaalajas toimivate veebirakenduste loomiseks.

Rakenduse ehitamiseks kasutati Node.js minimaalset Express raamistikku. Express on minimaalne ja paindlik veebirakenduste raamistik mis pakub veebirakenduste arendusele jõulisi iseärasusi [3]. Lõputööna valminud veebirakenduses valiti Express raamistik tema mugava renderdamisevõimaluse tõttu EJS failide puhul.

Rakenduse kliendipoolse kuvamiseks on implementeeritud EJS mall. EJS kombineerib andmed ning malli, et luua HTML fail [4]. EJS malli on võimalik serveri pool muutujatena kaasa anda andmeid (Joonis 1) ning kliendi pool kuvada <%= %> märgendi vahel (Joonis 2).

```
// NOTES
router.get('/notes', function(request, response) {
    var fetchScale = getRandomScale();
    var fetchFirst = shuffleNotesEasy(null, fetchScale);
    response.render(path.join(__dirname, '../public/views/notes/index.ejs'), {
        playNext : fetchFirst['note'],
        scale : fetchScale,
        dir: fetchFirst['dir']
    });
});
```

Joonis 1. EJS mallile andmete kaasaandmine

```
<script>var scale = "<%= scale %>";</script>
```

Joonis 2. Andmete kuvamine EJS mallis

Lisaks sellele on rakendusel andmevahetuse kergendamiseks kasutusel body-parser tehnoloogia. Body-parser on tehnoloogia, mis parssib Node.js poolt saadetud päringud vahevaras, muutes saadetud andmete kasutamise tunduvalt lihtsamaks ning mugavamaks.

Veebirakenduse kujunduse valikuks on Bootstrap raamistik. Bootstrap on kõige populaaresm HTML, CSS ja JS raamistik interaktiivsete ja mobiilsete projektide arendamiseks veebis [5]. Bootstrap annab kaasa HTML ja CSS programmeerimiskeeltel põhinevaid disainišabloone. Antud lõputöö raames anti Bootstrap kaasa välise CDN abil (Joonis 3).

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA712mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous">
</script>
```

Joonis 3. Bootstrapi lisamine

JavaScript funktsionaalsuse laiendamiseks on kasutatud jQuery teeki. jQuery lihtsustab JavaScript rakendamist veebirakendusel, kasutades JavaScript koodi ning ümbritsedes seda meetoditega, mis on välja kutsutavad vaid ühe rea koodi abil [6]. Antud lõputöö raames anti jQuery kaasa välise CDN abil (Joonis 4). jQuery teeki kasutati DOM manipulatsiooniks (Joonis 5).

```
<script src='http://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js'></script>
```

Joonis 4. jQuery lisamine

```
$('.has-error').addClass('has-success').removeClass('has-error');
$('.form-control-feedback').addClass('glyphicon-ok').removeClass('glyphicon-remove');
```

Joonis 5. jQuery DOM manipulatsioon

Interaktiivse kitarrikaela kuvamiseks kasutati kasutati vabavarana pakutavaid koodijuppe HTML, CSS ja JavaScript programmeerimiskeeltes [7]. Koodijupid on välja arendatud Andrey Pokrovskiy poolt ning kättesaadavad tema codepen keskkonnas [8].Vastavalt antud vabavara litsensi õigustele mis lubavad kolmandatel isikutel koodi muuta ja kasutada ilma teenustasudeta muudeti ja kassutati koodi HTML, CSS ja JavaScript faile, et implementeerida lõputöösse interaktiivne kitarrikael.

Veebirakenduse vasakpoolse menüü HTML mallina võeti aluseks valmis koodijupp, mis koosneb Bootstrap elementidest [9]. Lõputööna valminud veebirakenduses kasutati koodijupis leiduvale menüüle identset struktuuri ning elemente kuid menüüst jäeti välja otsingu võimalus.

3.3 Veebirakenduse struktuur

Veebirakendus põhineb klient-server mudelil (Joonis 6). Kliendile kuvatavad andmed valmistatakse ette rakenduse serveri poolel ning kliendi ja serveri vaheliseks andmevahetuseks kasutab antud veebirakendus HTTP edastusprotokolle.

Joonis 6. Klient-server mudel

Veebirakendusi arendatakase enamasti raamistike (*frameworks*) baasil. Raamistikud on lahendused, mida saab kasutada sageli korduvateks ülesanneteks [10]. Raamistike suureks boonuseks on valmiskomponentide ja struktuuri olemasolu. Lõputööna valminud veebirakenduses kasutatud Express raamistiku oluliseks valmiskomponendiks oli marsruutimine HTTP päringumeetodite jaoks.

Lõpuntööna valminud veebirakenduses kasutatakse kahte HTTP päringumeetodi: GET ja POST.

1) GET päringumeetodit kasutatakse EJS mallide kuvamiseks. Koos GET päringuga antakse vaadetesse playNext parameetrina kaasa esimene vastavalt mänguvormile noot või akord, mida kasutajale mängitakse mängu alustamisel (Joonis 7).

```
// CHORDS
router.get('/chords', function(request, response) {
    var playNext = shuffleChordsEasy();
    response.render(path.join(__dirname, '../public/views/chords/index.ejs'), {
        playNext : playNext
    });
});
```

Joonis 7. GET päring kus antakse playNext parameetrina kaasa esimene noot/akord

2) POST päringumeetodit kasutatakse erinevates vaadetes andmevahetuseks serveri ja kliendi vahel. AJAX tehnoloogia abil edastatakse vastavalt igale mänguvormi vaatele (Joonis 8) POST pärgingumeetodina serverisse päring kus on ära defineeritud päringu ja vastuse parameetrid (Joonis 9). Saadud parameetrid töödeltakse väljakutsutud funktsioonides ning tagastatakse JSON formaadis kliendile (Joonis 10).

```
$.ajax({
    type: 'POST',
    data: JSON.stringify(data),
    contentType: 'application/json',
    url: 'http://localhost:8080/chords',
    success: function(data) {
```

Joonis 8. AJAX tehnoloogia abil edastatav päring

```
// CHORDS POST
router.post('/chords', function(request, response) {
    playingLogicChords(request, response);
});
```

Joonis 9. POST päring serveris

```
    response.json({
        playNext: playNext,
        scoreCounter: score,
        successCounter: successCounter,
        played: request.body.played,
        responseMessage: responseMessage,
        directoryOf: directoryOf
    });
```

Joonis 10. Andmete tagastamine JSON formaadis

3.4 Veebirakenduse õppemeetod

Veebirakendus põhineb kompententuse õppemudelil. Kompetentsuse õppemudel (*Competency based learning)* [11] keskendub isiku võimekusele ning erinevalt traditsioonilisest õppemudelist tagab asjaolu, et isikut ei lasta edasi järgmisele tasemele kui antud isik ei ole täielikult läbi saanud eelmisest tasemest. Traditsioonilises õppemudelis (*Traditional teaching*) [12] keskendutakse grupi inimeste puhul kui antud

isikute tervikule võimetele kuid kompetentsuse mudelil süvenetakse iga üksikisku võimetele vastavalt.

Kasutajal pole võimalik harjutada raskemaid akorde ja noote kui kasutaja pole suutnud kompetentselt ära arvata süsteemi poolt algsel tasemel pakutud noodid/akordid. Antud veebirakendusel on rõhk pandud algaja taseme arendamiseks, mille tõttu on kompetentsuse õppemudel efektiivne valik.

Kompetentsuse õppemudel on sobilik nootide mänguvormi lisafunktsionaalsuse tõttu, mis fokusseerib kergema taseme puhul nootide mängimist samast helireast ning noodi äraarvamisel valib järgmise noodina eelmisest noodist kolme noodi vahemikus oleva noodi. Tänu sellele on algajal kasutajal lisaks oma muusikalise kuulmise arendamisele võimalus tutvuda ka heliridadega ning enese teadmata õppida heliridasid.

Kompetentsuse õppemudel laia rakendust akordide mänguvormi puhul ei leia, kuid samuti antakse algajal kasutajal võimalus rohkem tundma õppida algseid akorde, mis on kitarrimängus kõige olulisemad.

Kasutajal on järgmise harjutustaseme akordideni ja nootideni jõudmiseks kaks valikuvarianti:

1) Kasutaja täitis algselt vastavalt oma hinnangule kriteeriumid.

2) Kasutaja alahindas oma võimet algaja taseme puhul ning vastas õigesti vähemalt 5 korda järjest.

Rakenduse iga mänguvormi alustamisel küsitakse kasutajalt hinnang tema kompetentsuse kohta – valida on algaja ning edasijõudnu taseme vahel. Algaja tasemel on vaikimisi edasijõudmise kriteeriumiks sätestatud 10 õiget vastust ning edasijõudnu puhul on kriteerium 5 õiget vastust.

Nii algaja kui ka edasijõudnu taseme puhul on lisaks õigete vastuste kriteeriumitele süsteemi sisse ehitatud ka järjestikune õigete vastuste jälgimine parameetritena. Mõlema taseme puhul järgitakse kasutaja poolt järjestikuste õigete vastuste arvu, mille abil otsustatakse vajadust tasemevahetusele ennem algselt sätestatud kriteeriumite täitmist. Kui kasutaja vastab õigesti siis jälgimise parameetrit suurendatakse ühe võrra ning kui kasutaja vastas valesti siis seda parameetrit vähendatakase ühe võrra.

Kui kasutaja selgelt alahindas oma võimet ja valis algtasemeks algaja ning vastas järjest õigesti vähemalt 5 korda siis süsteem adapteerib ning hakkab koheselt küsima järgmisse taseme akorde ning noote. Järgmise taseme akorde küsitakse seni, kuni õigesti vastamise parameeter on suurem vähemalt arvust 4 või kui kasutaja on jõudnud oma hinnangus sätestatud punktisummani.

Samuti jälgitakse ka kasutaja ebaõnnestumisi. Kui kasutaja, sõltumata algtaseme valikust, pakub järjestikku pidevalt vale vastust muudetakse pärast noodi või akordi ära arvamist tema mängu raskustaset kergemaks. Antud muutus rakendatakse juhul, kui kasutaja järjestikuse vastamise parameetri suurus on väiksem arvust -5 ning muutus rakendatakse hoolimata sellest, kas kasutaja on oma hinnangule sätestatud punktisummani jõudnud või mitte. Järgmisele tasemele tagasi jõuab kasutaja taas siis kui tema järjestikuse vastamise parameetri suurus on suurem arvust -5.

Kriteeriumite parameetrid leiti lõputöö raames katsetamise meetodil. Katsetamisel peeti silmas huvi tekitamist kasutajas ning kasutaja valitud taseme hinnangu kinnitamist.

Algajale kasutajale peab andma võimaluse piisavalt aega lihtsamate akordide ning nootidega tutvumiseks milletõttu määratakse edasijõudmise kriteeriumiks 10 õiget vastust.

Edasijõudnu hinnangu valinud kasutaja puhul on oluline välja selgitada tema tegelik kompetents, mille tõttu määratakse edasijõudmise kriteeriumiks 5 õiget vastust selle asemel, et hakata kohe raskemaid akorde või noote küsima.

Taseme langus kui kasutaja järjestikuste vastuste parameeter on väiksem arvust -5 annab kasutajale piisavalt eksimusruumi ning tagab kindla kontrolli selgitamaks välja, kas kasutaja pakub oma vastust katse eksitus meetodil.

3.5 Veebirakenduse akordid ja noodid

Käesoleva lõputöö raames lindistati kõik veebirakenduses mängitavad akordid ning noodid lõputöö koostaja poolt elektroakustilisel kitarril. Akordide lindistamiseks kasutati iPhone 5S sisseehitatud mikrofoni, mille abil on võimalik helifaile lindistada .m4a kujul.

Nootide lindismisel võeti aluseks minoorsed pentatoonika heliread. Heliridadest lindistati A minoorne pentatoonika, B minoorne pentatoonika, C minoorne pentatoonika, D minoorne pentatoonika ning E minoorne pentatoonika.

Akordide lindistamisel võeti aluseks duurid, mollid ning 7. akordid. Akordidest lindistati A-duur, A-moll, A7, B-duur, B-moll, B7, C-duur, C-moll, C7, D-duur, D-moll, D7, E-duur, E-moll, E7, F-duur, F-moll, F7, G-duur, G-moll, G7.

Kokku lindistati 69 helifaili, millest 21 on akordide erinevad variatsioonid ning 48 on unikaalsed noodid.

3.6 Veebirakenduse kasutusjuhud

Veebirakenduse kasutamiseks tuleb kõigepealt vasakult menüüst valida kasutajale meelepärane mänguvorm. Mänguvormi valimisel suunatakase kasutaja vastavasse vaatesse. Igas vaates on kasutaja valitud mänguvormi lühikirjeldus, valimikulahter kasutaja hinnangu andmiseks ning „Start“ nupp mille vajutamisel käivitatakse mäng. „Start“ nupu vajutusel mängitakse kasutajale olenevalt mänguvormist ette esimene akord või noot.

Kombineeritud väljakutse ning akordide mänguvormi puhul on akordide ettemängimsel kaks sisendit: akord ja tema täpsustus. Oma vastuse saatmiseks peab kasutaja vajutama nuppu „Answer“. Vale vastuse puhul kuvatakse kasutajale teade „Try again!“ ning visuaalselt muudetakse kasutaja sisendi ümbrise värvus punaseks ja ette mängitakse eelnevalt mängitud akord. Õige vastuse puhul kuvatakse kasutajale teade „Correct answer!“ muudetakase kasutaja sisendi ümbris roheliseks ning kasutajale mängitakse ette järgmine akord.

Kombineeritud väljakutse ning nootide mänguvormi puhul on nootide ettemängimisel üks sisend: noot mis valitakse kautajale ettejoonistatud kitarriskaala pealt. Kitarriskaalal nootide peale vajutades muutub nende värv roheliseks mis indikeerib, et antud noot on valitud. Kitarriskaalal saab üles-alla liikuda „Go Up“ ja „Go Down“ nuppude abil. Pärast noodi valikut on kasutajal tarvis vajutada nuppu „Answer“ mille abil edastatakse kasutaja vastus. Vale vastuse puhul kuvatakase kasutajale teade „Try again!“ ning sama

nooti mängitakse uuesti. Õige vastuse puhul kuvatakase kasutajale teade „Correct answer!“ ning kasutajale mängitakse ette uus noot.

Kombineeritud väljakutse ning nootide mänguvormi puhul on nootide ettemängimisel kasutajale abistava tesktina kuvatud ka skaala, millelt noot mängiti.

Pärast igat vastamist kuvatakse kasutajale lisaks tema punktide koguhulk, mis annab teada palju kasutaja on õigesti vastanud.

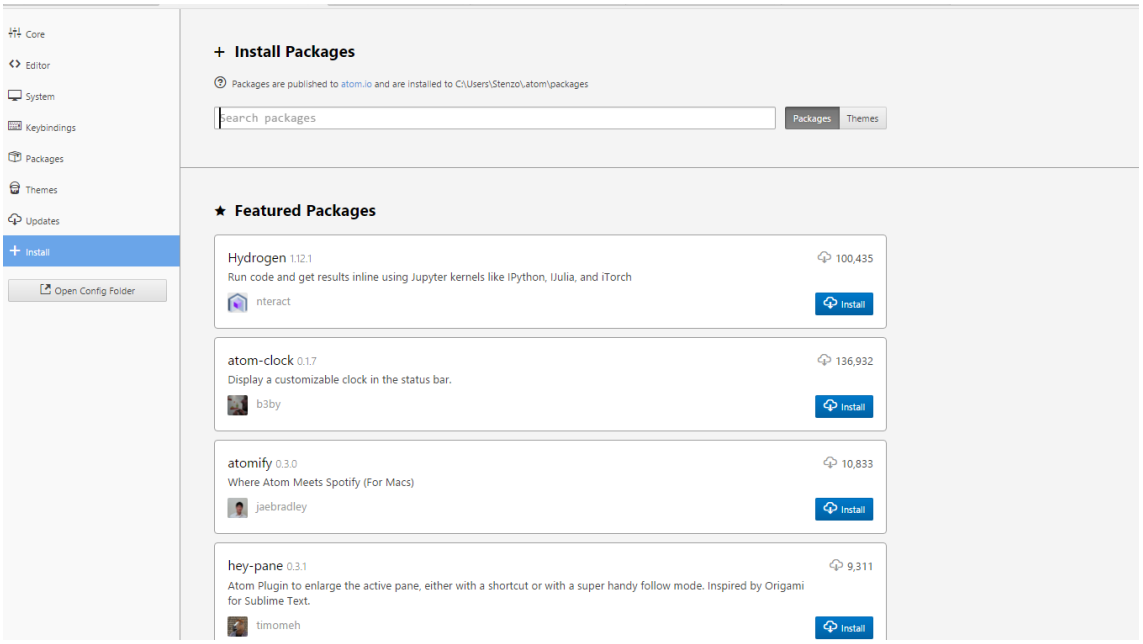
4 Kasutatud tööriistad

Järgnevas alampunktis tutvustatakse lõputööna valminud rakenduses kasutatud tööriistu.

4.1 Atom

Atom on avatud lähtekoodiga tekstiredaktor, mis on arendatud GitHub-i poolt. Atomisse on sisseehitatud git ning tugi lisamoodulitele. Enamus lisamooduleid on tasuta ning arendatud Atom kogukonna poolt. Atom betaversioon avalikustati 2015. aasta juunis. Atom on kirjutatud CoffeeScript ja Less programmeerimiskeelte abil.

Atom on suunatud põhiliselt veebirakenduste loomisele, toetades oma lisamoodulitega palju erinevaid programmeerimiskeeli. Atom lisamoodulite installeerimine on kerge ning moodulid on läbi liidese installeeritavad ning otsitavad (Joonis 11). Käesoleva töö raames kasutati veebirakenduse loomiseks Atom tekstiredaktorit. Valikuks osutus Atom tema lisamoodulite tõttu Node.js arendusel ning oma kasutajasõbraliku liidesse tõttu.



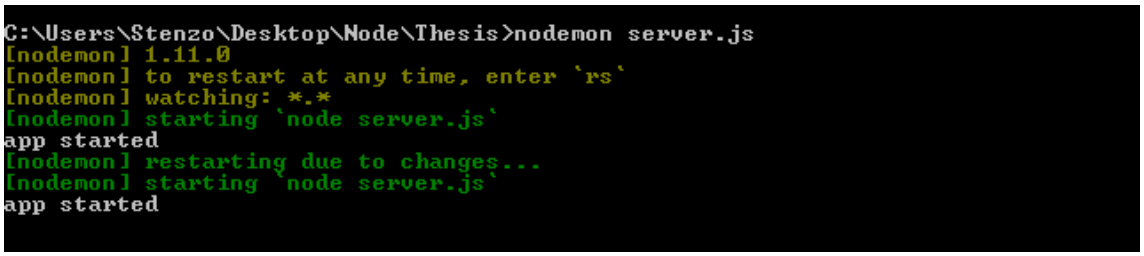
Joonis 11. Atom lisamoodlite otsimine

4.2 Nodemon

Nodemon on monitooringuvahend, millega jälgitakse lähtekoodis koodimuutust ning mille abil käivitatakse Node.js server. Nodemon on kasulik abivahend Node.js arendusel, sest vastavalt koodimuutusele taaskäivtab Nodemon Node.js serveri (Joonis 12). Nodemon on avatud lähtekoodiga ning installeeritav läbi terminali.

```
npm install -g nodemon
```

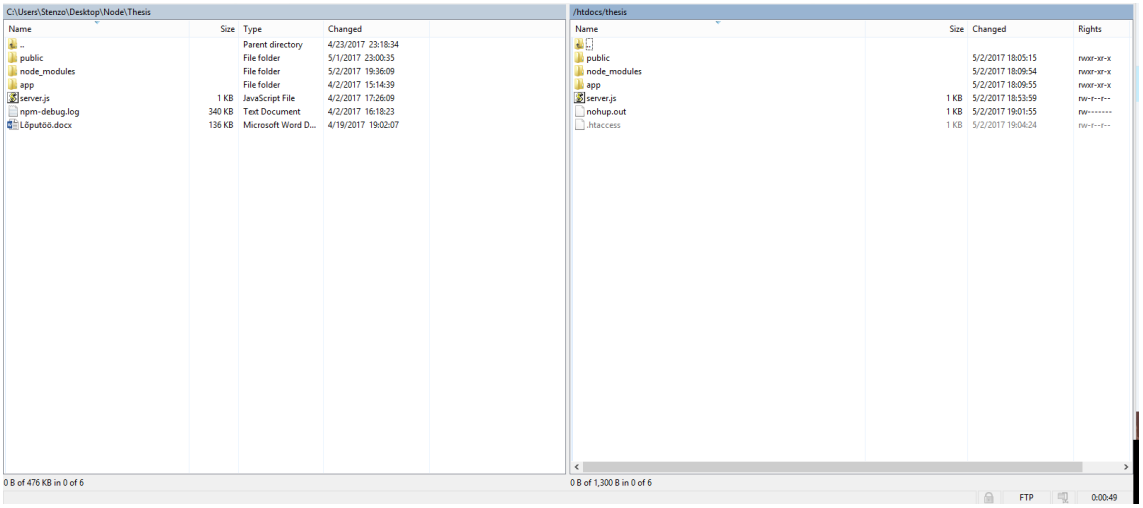
Nodemon installeerimine käsurealt



Joonis 12. Automaatne taaskäivitus pärast koodimuutust

4.3 WinSCP

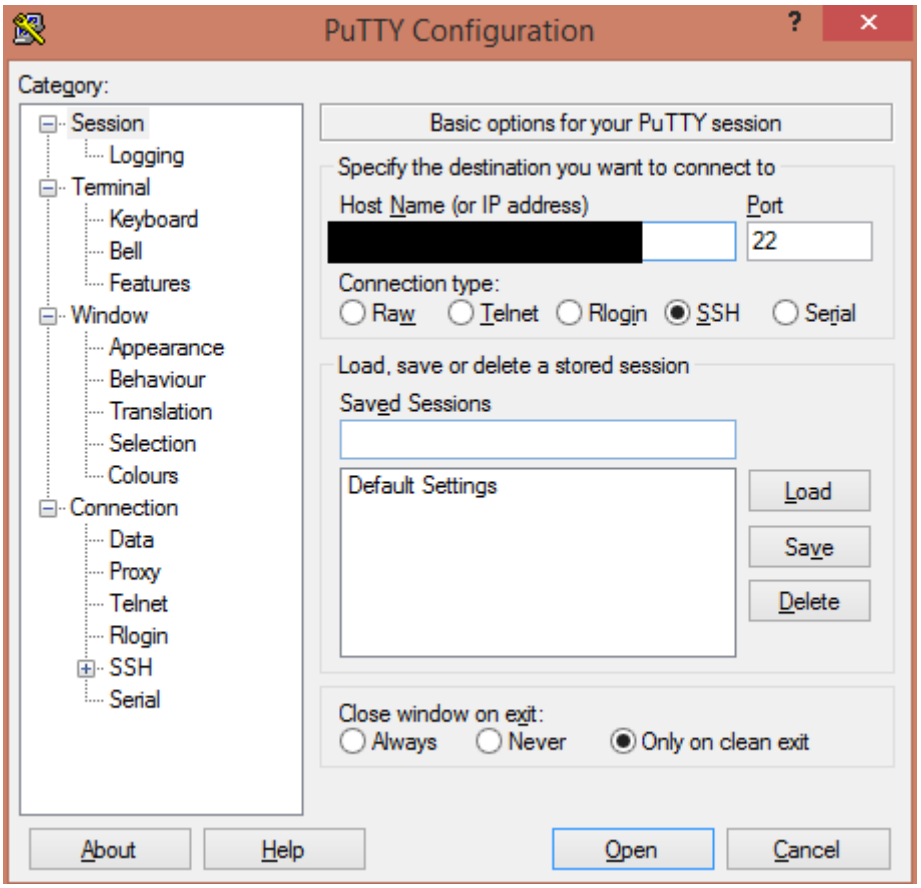
WinSCP on avatud lähtekoodiga ning tasuta SFTP ja FTP klient Windows operatsioonissüsteemi jaoks. WinSCP funktsionaalsuseks on mugav ning turvaline andmevahetus kohaliku masina ning serveri vahel [13]. Antud lõputöö raames kasutati WinSCP failide ületõstmiseks kohalikust masinast veebiserverisse. WiNSCP abil on võimalik faile suurtes kogustest ning mahtudes kergesti oma kohalikust masinast kopeerida ning tõsta üle serverisse (Joonis 13).



Joonis 13. WiNSCP liides, vasakul kohalik masin, paremal server

4.4 PuTTY

PuTTY on populaarne SSH ning Telnet klient, mis aitab luua turvalisi ühendusi üle interneti [14]. Käesoleva lõputöö raames kasutati PuTTY klienti SSH ühenduse loomiseks serveriga ning PuTTY käsurida Node.js moodulite installeerimiseks ning Node.js käivitamiseks veebimajutuse serveri poolel. PuTTY abil on kergesti võimalik sätestada parameetrid SSH ühenduse jaoks serveriga (Joonis 14).



Joonis 14. PuTTY kasuajaliideses parameetrite sätestamine serveriga ühendamiseks

5 Hinnang

Järgnevas alampunktis antakse lõputöö valmistaja poolt hinnang lõputööna valminud veebirakendusele.

Veebirakenduse arenduse plaanitud alguskuupäevaks oli 1. märts ning planeeritud lõpuks 1. mai. Tegelikkuses alustati arendamist 10 märtsil ning veebirakendus valmis 10. mail.

Vastavalt arenduse algsele plaanile pidi kolmanda mänguvormi nimetuseks olema ajaline väljakutse. Antud mänguvormis oli esitatud eesmärk üles ehitada mänguvorm, kus mängitakse kasutajale ette akorde ning noote teatud aja vältel ning aja lõppedes oleks pidanud kasutaja saama salvestada enda tulemue andmebaasi. Algselt leidis ajalise väljakutse mänguvorm ka rakendust, kuid arenduse hilisemas faasis seati kahtluse alla nimetatud mänguvormi otstarbekus ning idee algsest teostusest loobuti. Selle asemel tehti kolmandaks mänguvormiks kombineeritud mänguvorm, kus küll mängitakse nii noote kui ka akorde, kuid puudub võimalus oma tulemuse salvestamiseks andmebaasi ning puudub aja jälgimine.

Lisaks tekkis arenduse käigus ka uusi ideid kasutajaliidese arendamiseks. Nootide mänguvormi puhul pidi algselt olema võimalus sisestada numbrina sisendisse, kuid arenduse käigus leiti huvitav uus viis oma sisendi sisestamiseks – interaktiivne kitarrikael. Interaktiivne kitarrikael implementeeriti kombineeritud väljakutse noodi ettemängimisel ning nootide mänguvormi jaoks.

Veebirakendus on mõeldud nii arvuti kui ka mobiilivaate jaoks. Liidesel on vastavalt sellele kolm olekut:

1) Lauaarvuti vaade kui ekraani laius on vähemalt 768 pikslit.

2) Tahvelarvuti vaade kui ekraani laius on 768 ja 632 piksli vahemikus.

3) Mobiilivaade kui ekraani laius on alla 632 piksli.

Kuna veebirakenduses kasutatud interaktiivne kitarriskaala nõudis väga palju kujunduse muutmist ning antud kitarr ei toetanud interaktiivset kujundust siis mobiilivaates ei ole antud kujundus täielikult interaktiivne ning mobiilivaade on osaliselt vigane.

Lõputööna valminud veebirakendus anti testimiseks pianistile, kes on õppinud muusikateooriat ning kitarrimängu. Antud isik leidis nootide mänguvormi puhul, et interaktiivne kitarriskaala on väga huvitav viis oma vastuse sisestamiseks. Akordide mänguvormi puhul tundus vastuse sisestamine üksluine ning ka mõnel määral tüütu. Hinnang antud isiku poolt loodud veebirakenduse vajadusele oli positiivne, sest muusikaline kuulmine on oluline oskus isikutele, kes mängivad instrumenti.

Lõputöö esitaja on rahul funktsionaalsusega, mis hõlmab nootide ettemängimist ning nootide vastuse sisestamist. Implementeeritud algoritm ning adapteerimine on efektiivsed peidetud lisafunktsionaalsused, mis aitavad veebirakenduse kasutajal lisaks muusikalise kuulmisele arendada ka enda teadmisi heliridadest.

Esitaja arvates oleks võinud paremini valmistada akordide sisendiga ning mänguvormiga seonduva. Probleemiks osutus parima akordide sisendmeetodi loomine kuna hetkel lõputöös implementeeritud lahend ei muuda veebirakendust unikaalseks ning kitarrikaela rakendamine akordide puhul oleks muutunud kasutajavaenulikuks – iga akordi puhul oleks vaja kasutajal teha kuni kuus valikut nootideks.

Kõik ülejäänud funktsionaalsused ning nõuded, mis püstitati on täidetud ning töötavad vastavalt püstitusele ja on ka kergesti edasiarendatavad.

6 Kokkuvõte

Järgnevas alampunktis esitatakse kokkuvõte lõputööna valminud veebirakenduse jaoks.

Käesolevas töö põhieesmärgiks oli luua funktsionaalne veebirakendus, mis aitab kasutajal arendada tema muusikalist kuulmist. Veebirakenduse eesmärk oli rakendada kolme mänguvormi: akordid, noodid ning kombineeritud väljakutse.

Lõputöö eesmärk oli veebirakendus üles ehitada Node.js platformil, mille jaoks kasutatakse JavaScript programmeerimiskeelt. Rakenduse kliendipoolsete vaadete kuvamiseks kasutati EJS malli ning kujundus tehti Bootstrap põhjal.

Veebirakenduse sisse on kavandatud õppemudel ning algoritm, mille abil määratakse veebirakenduse kasutajale sobiv mängutase.

Töö tulemuseks on funktsionaalne veebirakendus, mille abil on kasutajal võimalik arendada muusikalist kuulmist kitarri abil. Antud veebirakendust on võimalik edasi arendada ning lisada lisafunktsionaalsusi.

Kasutatud kirjandus

[1] What is a Web Application? [WWW]

https://www.maxcdn.com/one/visual-glossary/web-application/ (04.04/2017)

[2] Veebimajutus [WWW]

https://www.veebimajutus.ee/ (30/04/2017)

[3] Node.js – Express Framework [WWW]

https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm (30/04/2017)

[4] EJS The Data [WWW]

http://www.embeddedjs.com/ (30/04/2017)

[5] Bootstrap [WWW]

http://getbootstrap.com/ (30/04/2017)

[6] jQuery introduction [WWW]

https://www.w3schools.com/jquery/jquery_intro.asp (30/04/2017)

[7] Guitar Fretboard Visualization [WWW]

https://codepen.io/DreySkee/pen/bddpqM (02/05/2017)

[8] Andrey Pokrovskiy [WWW]

https://codepen.io/DreySkee/ (02/05/2017)

[9] W3Schools [WWW]
https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_temp_blog&stacked=h (02/05/2017)

[10] Veebirakendused [WWW]

https://www.ria.ee/public/ISKE/Veebirakendused.pdf (19.04/2017)

[11] Competency-Based Learning or Personalized Learning [WWW]

https://www.ed.gov/oii-news/competency-based-learning-or-personalized- learning (06/05/2017)

[12] Non-Traditional Teaching & Learning Strategies [WWW]

http://www.montana.edu/facultyexcellence/Papers/activelearn2.html (06/05/201)

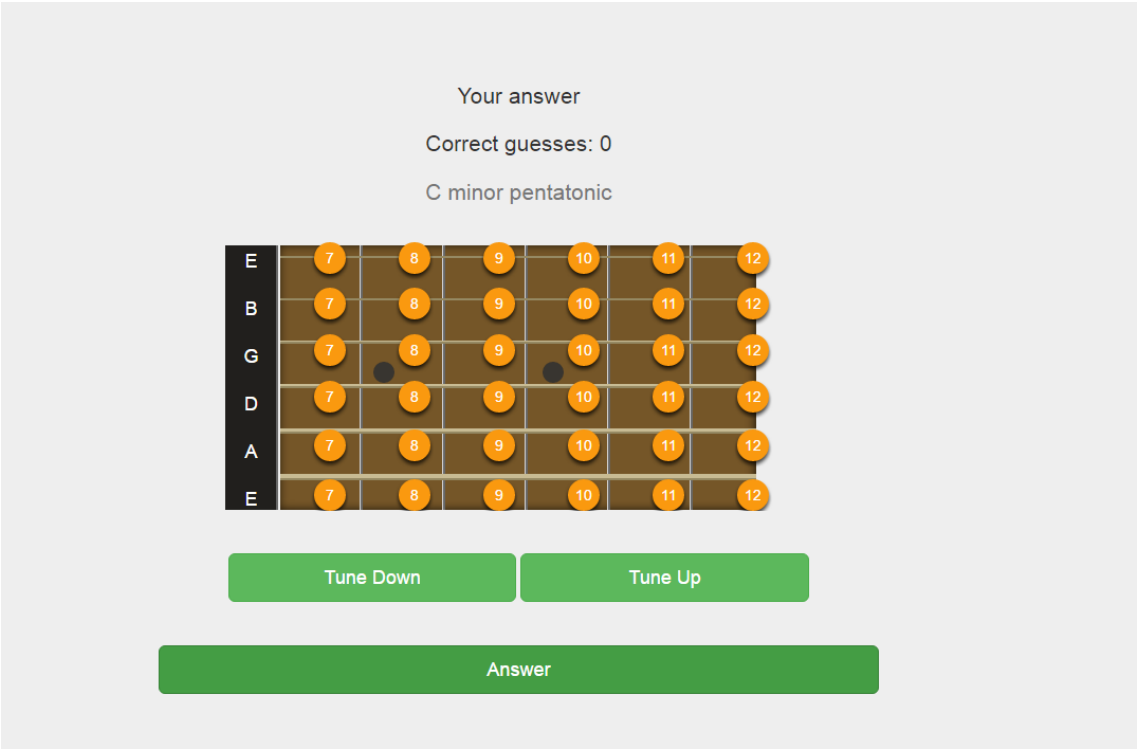
[13] WinSCP [WWW]

http://filehippo.com/download_winscp/ (02/05/2017)

[14] PuTTY [WWW]

https://putty.en.softonic.com/ (02/05/2017)

Lisa 1 – nootide mänguvormi vaade



Lisa 2 –akordide mänguvormi vaade