SZAKDOLGOZAT

Csizmazia Máté   
2020

Pannon Egyetem   
Műszaki Informatikai Kar  
Rendszer- és Számítástudományi Tanszék   
Programtervező informatikus BSc

SZAKDOLGOZAT

Webalapú szintetizáló és audio produkciós szoftver

Csizmazia Máté

Témavezető: Szekér Szabolcs

2020

>Feladatkiírás<

Csere az aláírt témakiírásra!

Nyilatkozat

Alulírott *Csizmazia Máté* hallgató, kijelentem, hogy a dolgozatot a Pannon Egyetem *Rendszer- és Számítástudományi* tanszékén készítettem az okleveles programtervező informatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozatban lévő érdemi rész saját munkám eredménye, az érdemi részen kívül csak a hivatkozott forrásokat (szakirodalom, eszközök stb.) használtam fel.

Tudomásul veszem, hogy a dolgozatban foglalt eredményeket a Pannon Egyetem, valamint a feladatot kiíró szervezeti egység saját céljaira szabadon felhasználhatja.

Veszprém, 2020. május 10.

Csizmazia Máté

Alulírott *Szekér Szabolcs* témavezető kijelentem, hogy a dolgozatot *Csizmazia Máté* a Pannon Egyetem *Rendszer- és Számítástudományi* tanszékén készítette okleveles programtervező informatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozat védésre bocsátását engedélyezem.

Veszprém, 2020. május 10.

Szekér Szabolcs

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani mindazoknak, akik szakmailag és erkölcsileg hozzájárultak a szakdolgozatom megvalósításához. Kitartó munkájuknak, segítőkészségüknek és szellemi támogatásuknak köszönhetően jöhetett létre a dolgozat.

Köszönöm témavezetőmnek, Szekér Szabolcsnak a sok segítséget, támogatást és építő kritikát. Munkája és szakmai hozzáértése nélkül nem születhetett volna meg ez a dolgozat és a hozzá tartozó szoftver.

Továbbá hálával tartozom családomnak a kitartó szellemi támogatásért és azért, hogy lehetővé tették tanulmányaimban való előre haladásom.

Külön köszönetet szeretnék mondani Bödey Ádámnak, aki szakmai tapasztalatával segítette a webes megvalósítás biztos és megbízható működését.

Tartalmi összefoglaló

Az audio szintetizáló programok esetében gyakori probléma a hanggenerálás működésének hiányos ismerete. Ezen szoftverek használatának esetében, túlnyomó többségben nem vagyunk képesek megérteni a folyamat menetét, és hogy mit befolyásolunk mennyire. Érdemleges eredményeket csak akkor tudunk elérni ezeknél szoftvereknél, ha tisztában vagyunk az audio szintézis és effekt használat alapjaival.

Egy zenei kompozíció elkészítésekor nem elég figyelembe venni a dallamot vagy a harmóniát, mivel ez nem elég ahhoz, hogy egy hangzatos művet készítsünk. Megfelelő háttértudás nélkül, nincs lehetőségünk elkészíteni azt a hangzást amire vágyunk, vagy a fejünkben összeállt.

Dolgozatomban egy olyan webalapú szoftvert mutatok be ami a webalapú szintetizáló, és audio produkciós szoftver névre hallgat, amelyben lehetőség nyílik megismerni az audio szintézis fajtáit, audio effektek alkalmazásának módszereit, és ezzel egyidőben lehetőséget ad audiovizuális feedback-re, aminek a segítségével sokkal egyszerűbb a téma megértése. A szoftver elkészítéséhez a webfejlesztésben nélkülözhetetlen JavaScript-et alkalmazom, aminek a backend felülete NodeJS technológiát alkalmaz, frontend felülete pedig Bootstrap-et. Egy ToneJS nevű JavaScript keretrendszer segítségével lehetőség nyílik a hanggenerálásra, és annak részletes bemutatására. Ez a keretrendszer képes effektek használatára, sequencing / timing metódusokra, vizualizációra, és még számos funkcióra, ezért nélkülözhetetlen ennek a használata. Ezen felül, a szoftver, tanuló felületként is működik, a könnyebb, gördülékenyebb tudásátadás érdekében.

Kulcsszavak: Node, JavaScript, ToneJS, Szintetizátor, Web

Abstract

A common problem with audio synthesizer programs is the lack of knowledge of how sound generation works. The vast majority of these softwares we create are just attempts and can not help us understand the process and how much we influence the generated sound. Only if we understand the basics of audio synthesis and effect use can we achieve satisfying results.

When creating a musical composition, it is not enough to take into account the melody or the harmonies, because it is not enough to make a musical work. Without proper background knowledge, there is no way to produce the sound that we desire.

In my dissertation I present a web-based software that gives an opportunity to learn about the types of audio synthesis, methods of applying audio effects, and at the same time gives an opportunity for audio-visual feedback that makes it much easier to understand the topic. I use JavaScript in web development, which uses NodeJS technology for back-end and Bootstrap for the front-end. Using a JavaScript framework called ToneJS, it is possible to generate and present audio in detail. This framework is capable of using effects, sequencing / timing methods, visualization and many more functions, so it’s use is essential. In addition, the software works like a tutorial, for easier, smoother learning.

Key words: Node, JavaScript, ToneJS, Synthesizer, Web

Tartalomjegyzék

[1. Bevezetés 1](#_Toc39760546)

[2. Jelfeldolgozás 3](#_Toc39760547)

[2.1. DSP (Digital Signal Processing) rendszerek 3](#_Toc39760548)

[2.1.1. Felépítésük [3] 3](#_Toc39760549)

[2.2. Gyors Fourier transzformáció (Fast Fourier Transform, FFT) 6](#_Toc39760550)

[2.3. Szűrők (Filterek) [4] 6](#_Toc39760551)

[3. Hangszintézis 8](#_Toc39760552)

[4. Versenytárs-elemzés 10](#_Toc39760553)

[4.1. MIDI.CITY 10](#_Toc39760554)

[4.2. Xfer Serum / SerumFX 12](#_Toc39760555)

[5. Technológiák 14](#_Toc39760556)

[5.1. Webfejlesztés 14](#_Toc39760557)

[5.2. Front-end 15](#_Toc39760558)

[5.2.1. EJS 16](#_Toc39760559)

[5.2.2. Bootstrap 17](#_Toc39760560)

[5.2.3. NexusUI 18](#_Toc39760561)

[5.3. Back-end 20](#_Toc39760562)

[5.3.1. Node.js 20](#_Toc39760563)

[5.3.2. Express 21](#_Toc39760564)

[5.3.3. Nodemon 22](#_Toc39760565)

[5.4. Web technológiai alapok 22](#_Toc39760566)

[5.4.1. Routing 22](#_Toc39760567)

[5.4.2. Metódusai 23](#_Toc39760568)

[5.4.3. Státuszkód 24](#_Toc39760569)

[5.4.4. Middleware 24](#_Toc39760570)

[5.5. Tone.js 25](#_Toc39760571)

[5.6. MongoDB 28](#_Toc39760572)

[5.6.1. Mongod 29](#_Toc39760573)

[5.6.2. Mongoose 29](#_Toc39760574)

[5.6.3. Passport 29](#_Toc39760575)

[6. Célkitűzés 31](#_Toc39760576)

[7. Megvalósítás 33](#_Toc39760577)

[7.1. Szerver 33](#_Toc39760578)

[7.2. Kezdő oldal 34](#_Toc39760579)

[7.3. Autentikáció 35](#_Toc39760580)

[7.4. Tanuló felület 36](#_Toc39760581)

[7.5. Szintetizátor 37](#_Toc39760582)

[7.6. Mentett elemek felülete 39](#_Toc39760583)

[8. Továbbfejlesztési lehetőségek 41](#_Toc39760584)

[8.1. Hangsávok 41](#_Toc39760585)

[8.2. Projekt megosztás 41](#_Toc39760586)

[8.3. Kollaboráció 41](#_Toc39760587)

[8.4. Automatizáció 41](#_Toc39760588)

[8.5. Dob variációk 42](#_Toc39760589)

[8.6. Hangszerek 42](#_Toc39760590)

[8.7. MIDI 42](#_Toc39760591)

[8.8. Exportálás 43](#_Toc39760592)

[9. Összefoglalás 44](#_Toc39760593)

[*Irodalomjegyzék* i](#_Toc39760594)

[*Ábrák jegyzéke* ii](#_Toc39760595)

[*Táblázatok jegyzéke* iii](#_Toc39760596)

[*CD Melléklet* iv](#_Toc39760597)

# Bevezetés

Már említésre került a tartalmi összefoglalóban, hogy az audio szintetizáló programok esetében gyakori probléma a hanggenerálás működésének hiányos ismerete. Ezen szoftverek esetében, amiket létrehozunk hangok és a hozzá tartozó beállítások, túlnyomó többségben csak próbálkozások, nem segítenek megérteni a folyamat menetét, és hogy mit befolyásolunk mennyire. Érdemleges eredményeket csak akkor tudunk elérni ezen szoftverek esetében, ha tisztában vagyunk az audio szintézis és effekt használat alapjaival.

Egy zenei kompozíció elkészítésekor nem elég figyelembe venni a dallamot vagy a harmóniákat, mivel ez nem elég ahhoz, hogy egy hangzatos művet készítsünk. Megfelelő háttértudás nélkül nincs lehetőségünk elkészíteni azt a hangzást, amire vágyunk vagy a fejünkben összeállt. Amíg a zenei állomások használata egy bonyolult feladat az összetett elrendezés és számos funkciója miatt, addig ezen szoftverek ára is kiemelkedően magas. Egy átlag felhasználó, aki még nem jártas a zenekészítésben, nem szívesen költene magas összegeket arra, hogy kipróbálhassa magát e területen. Ezért a szoftver használata teljesen ingyenes, és bárki számára elérhető, aki rendelkezik egy web böngészővel. A szoftver elkészítésénél törekszem a lehető legletisztultabb, és egyértelmű működést biztosítani a felhasználónak, ezzel egy kiemelkedő élményt nyújtani.

Dolgozatomban egy olyan webalapú szoftvert mutatok be, amelyben lehetőség nyílik megismerni az audio szintézis fajtáit, audio effektek alkalmazásának módszereit, és ezzel egyidőben lehetőséget ad audiovizuális feedback-re, aminek a segítségével sokkal egyszerűbb a téma megértése. A szoftver elkészítéséhez a webfejlesztésben nélkülözhetetlen JavaScript-et alkalmazom, aminek a backend felülete NodeJS technológiát alkalmaz, frontend felülete pedig Bootstrap-et. Egy ToneJS nevű JavaScript keretrendszer segítségével lehetőség nyílik az hanggenerálásra, és annak részletes bemutatására. Ez a keretrendszer képes effektek használatára, sequencing / timing metódusokra, vizualizációra, és még számos funkcióra, ezért nélkülözhetetlen ennek a használata. Ezen felül a szoftverben lehetőség nyílik a zenekészítés, és effekthasználat alapvető megismerésére, így kedvet hozva a felhasználónak a téma elsajátításában. A szoftver azokat a felhasználókat célozza meg, akik még nem rendelkeznek a témában semmilyen, vagy csak minimális tudással, de a már jártas végfelhasználók is több módon szórakozónak találhatják az elkészített alkalmazást.

# Jelfeldolgozás

## DSP (Digital Signal Processing) rendszerek

A modern elektronikának a jelenleg is fejlődő ága. Ezen rendszerek felhasználása megfigyelhető mind a szórakoztató iparban, a szabályozástechnikában és az informatikában. Használata nélkülözhetetlen, mivel bármi olyan eszközről beszélünk, ami analóg jelet fogad, annak át kell alakítani ezt egy digitális jellé. [1,3]

Analóg jel

Az analóg jelet általában folytonos fizikai mennyiség határozza meg, amely folyamatos függvénye az időnek, helynek vagy akár más egységnek is.

Digitális jel

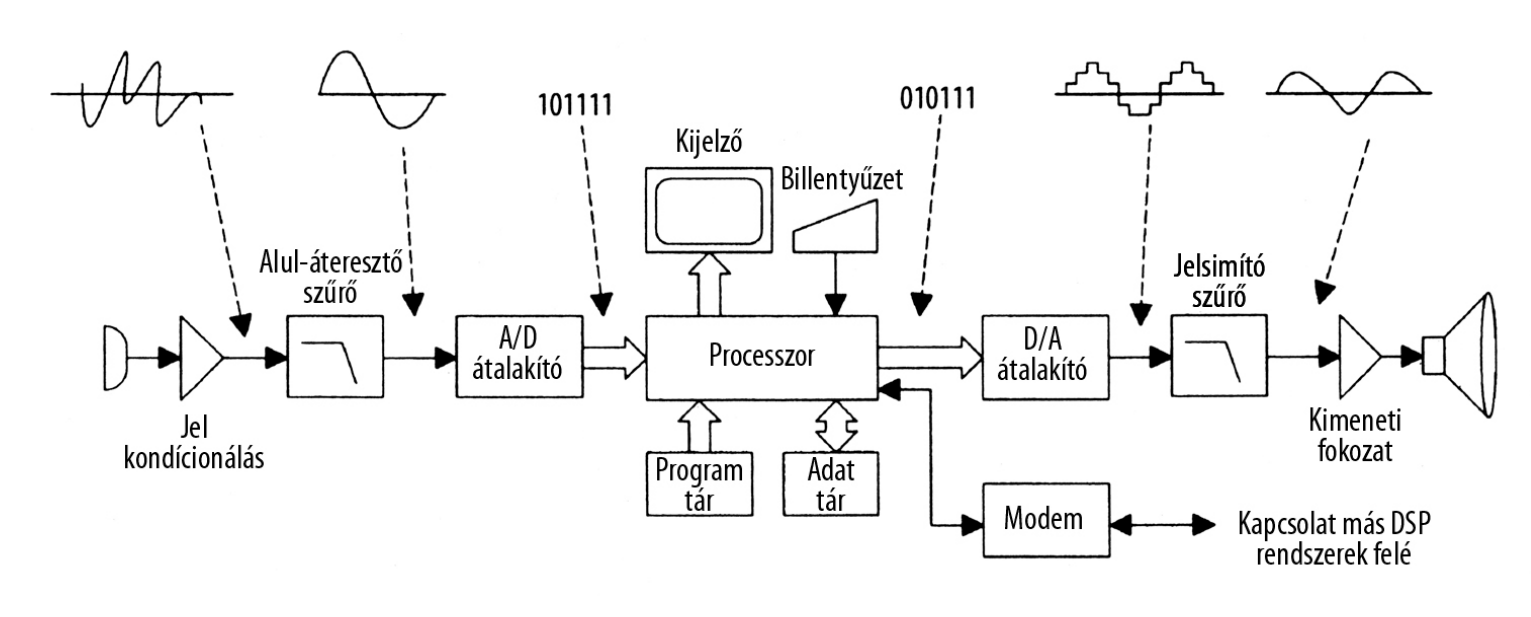
A digitális jel, egy diszkrét számokkal kódolt (speciális bináris) megjelenítési forma. Diszkrét jel alatt azt értjük, hogy a jelek véges sok könnyen megkülönböztethető értékeket és formákat vehetnek fel. [2]

Tehát a fő eltérés a két jel között a folytonos és a diszkrét idő.

Amíg mi ezen jelek között nem halljuk/látjuk a különbséget, a DSP rendszerek megállás nélkül dolgoznak, hogy a jel az emberi fülhöz és szemhez el tudjon jutni, mint információ. [2]

### Felépítésük [3]

A következő ábrán egy DSP rendszer modellje látható, ami segít majd ennek működése megértésében. Az alábbi csoportokba rendezhetőek a rendszer segítségével végrehajtott operációk:



1. ábra Egy digitális analóg jelátalakító ( DSP ) egyszerű modellje [3]

* Analóg jelfogadás az input csatornán keresztül
* Ezeknek a jeleknek számokká alakítása és ábrázolása
* A kapott értékhalmazon funkciós számítások elvégzése
* Végül ezen számok konvertálása vissza, analóg jellé, jelekké

Eközben kerül feldolgozásra az az információ, amit - tárolhatunk, megjeleníthetünk, vagy akár továbbíthatunk is. Ez a modell csak általánosan mutatja be az ilyen rendszerek működési folyamatát. A megoldások szétszedhetők külön hasonló részekre, de ezek működése a szoftvertől vagy használt technológiától is jelentősen függ.

A bemenet és a jelérzékelő

Az összes beérkező jelet (fizikai vagy kémiai) ennek az eszköznek át kell alakítania elektromos jellé. Ilyenkor ez egy olyan jeltartományba képződik le, ami folytonos, és egységes, amely egy előre definiált pontosság alapján követi a paraméter változásokat, és egy adott pillanatban mért értékét. Ilyen eszközök például egy antenna, vagy egy mikrofon.

Jelrendezés és simítás

Mivel a beérkező jel nem feltétlen csak olyan értékeket vesz fel, ami az eszköz megfelelő tartományába esik, ezért ilyenkor a bemeneti jelet úgy igazítja, hogy a további részek is számára biztos működést tudjon szolgáltatni. Általában erősítők és leválasztók alkotják a rendszer ezen részét.

Anti-aliasing szűrés

Ez a része a rendszernek, egy alul-áteresztő filter. Az analóg digitális átalakító számára biztosítja használható sebességét a jelnek, úgy, hogy korlátozza a jel változásának sebességét. Azért fontos, hogy a DSP képes legyen követni a bemeneti jelet és annak változását, mivel túl gyakori változással képtelen lenne feldolgozni ezt.

Analóg-digitális átalakító

Ezek az eszközök a felelősek a jelek számokká, értékekké alakításáért, tehát egy jól meghatározott értékelési rendszerbe alakítják át, digitalizálják. Egy olyan tartományba továbbítják a jelet, hogy az értelmes, és feldolgozható legyen a többi rendszer számára is. Az átalakítóknak a főbb jellemzői közé sorolható a konverziós sebesség, ami a használt módszer sebességét jelzi és a finomsága a felbontásnak, ami megmondja, hogy a diszkrét értékek mennyire közeliek a valósághoz.

Processzor

Az algoritmus megvalósítása tartozik a főbb feladati közé, az előző fokozatból nyert adatok alapján. A rendszer működéséhez nélkülözhetetlen alkatrész.

Program- és adattárolás

Harvard architektúra alapján, külön memóriában vannak eltárolva, az adatok, és külön a program is. Nem véletlen, hiszen ezáltal a működését felgyorsítja, mivel utasítások sorozata alatt a kód beolvasása és feldolzása alatt külön buszokon tudja beolvasni, vagy továbbítani az adatokat.

Adattovábbítás

Előnyt jelent az analóg jelekkel szemben, a digitális jelek tárolása/továbbítása, mivel a digitalizált jeleket egyszerűen, a megfelelő eszközön hosszú ideig, bármiféle kockázat nélkül tárolhatók és végtelenszer felhasználhatók.

Digitális-analóg átalakító

Számos DSP létezik analóg kimenettel is, ezért ezen digitális adatokat szükséges visszaállítani elektromos formában, feszültséggé vagy áramjellé, amit később további rendszerekhez tudunk eljuttatni.

Kimeneti simítás

Az digitális analóg átalakító után kimenő jelek jelentősen „szögletesek”, amik a jeltovábbítás és felhasználási területen zavarokat okozhatnak, ezért ezek simítására van szükség, egy alul-áteresztő szűrő segítségével.

## Gyors Fourier transzformáció (Fast Fourier Transform, FFT)

A gyors Fourier transzformáció (Fast Fourier Transform, FFT) egy olyan algoritmus, aminek a segítségével kiszámíthatjuk a diszkrét Fourier transzformáltat. Megvalósításánál mintaesetén megközelítőleg műveletet kell elvégezni, amely kifejezetten hosszú időt vesz igénybe, ilyen esetekben alkalmazható az FFT mivel sebességét tekintve jelentősen gyorsabb algoritmus. [3]

## Szűrők (Filterek) [4]

A szűrőknek rengeteg fajtája lehet: alul áteresztő (low-pass), felül áteresztő (high-pass), sáváteresztő szűrő (band-pass). Ezeket rengeteg módon alkalmazhatjuk, mivel nem csupán hangzatos eredményeket kaphatunk, hanem hasznosat is. Például a mély hangszín szabályozó a sztereó hangrendszereken a jel mély/alacsony frekvenciáinak az amplitúdóit, a magas hangszín szabályozó pedig a magas frekvenciáinak az amplitúdóit alakítja át. Kiemelhetjük az eltérő frekvenciájú hang jeleket. Kettő általános szűrő segítségével csökkenthető a zaj és csonkítható a sávszélesség. Egy alul áteresztő szűrő biztosításával, csökkenti/csonkítja a mintavételezést.

Ezen szűrési folyamat következtével megváltoznak a jelet alkotó frekvenciák amplitúdói, és lehetővé teszi, hogy a számunkra lényeges részeket kiemeljük a zajos jelből.

A digitális szűrok viszont számos előnnyel rendelkeznek az analóggal szűrőkkel szemben:

* Pontosabb.
* Olyan szűrők is megvalósíthatók, amelyek analóggal nem.
* Paraméterei programozhatók.
* Gyorsan tesztelhetők.
* Egyszerű számtani műveletekkel dolgoznak.
* Jó a teljesítmény/költség aránya.
* Nem befolyásolják a környezeti tulajdonságok (hőmérséklet, páratartalom)
* Nem öregednek el az analóggal szemben.

# Hangszintézis

Ez a leglényegesebb része a szakdolgozatomnak, mivel erre alapszik a nagy része a szoftvernek. Hanggenerálást bármilyen programozási nyelvben létre tudunk hozni a megfelelő függvények és könyvtárak segítségével. A dolgozatban ezt a funkciót a ToneJS nevű JavaScript könyvtár segíti a megfelelő működésben.

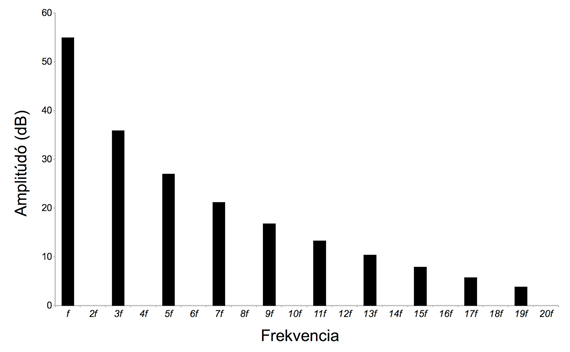
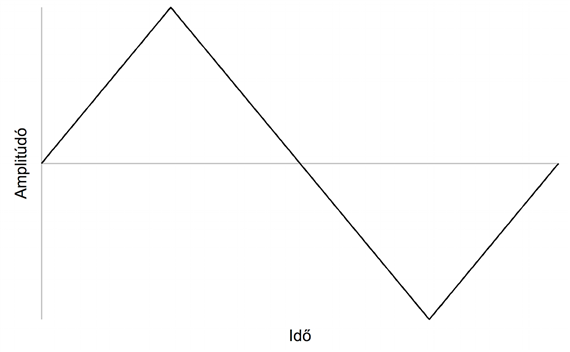
***Elmélet***

Hanggenerálás során először szükségünk van meghatározni a létrehozandó hang egy periódusát (hullámformát), majd ezt a létrehozott struktúrát ismételjük. Az így megkapott hang frekvenciáját az ismétlődés sebessége határozza meg (vagyis, az a hullámformát egységnyi idő alatt hányszor játsszuk le), míg a hangszínt a hullámforma alakítja.

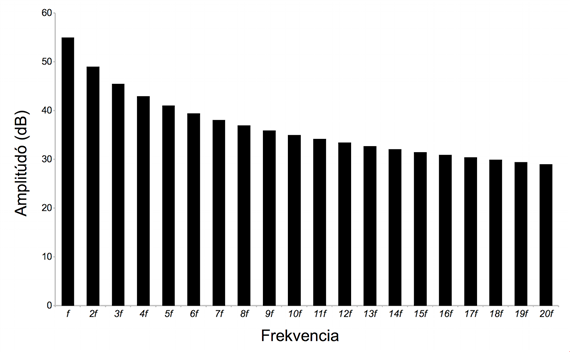
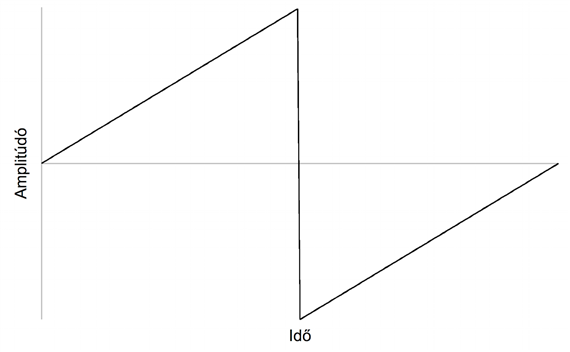
A generált hullámformák száma végtelen, de általában négyet nevezünk meg amikor az alap hullámformákról (waveform) beszélünk:

* szinusz (sine),
* háromszög (triangle),
* négyszög (square),
* fogazott (sawtooth)

Ezeknek a hullámformáknak egyértelműen hangzásbéli eltérései vannak, amik egyszerűen észrevehetők. A szinusz hullámnak jellegzetes, tiszta hangzása van, ennek köszönhetően leggyakrabban a mély/basszus elemekbe építik be (ez nem azt jelenti, hogy más esetekben nem használhatók), ez magyarázható azzal, hogy a frekvenciaspektrumon nincs akkora jelenléte, mint mondjuk egy fogazott hullámformának.



2. ábra Egy háromszög hullámforma által generált frekvencia értékeket leíró hang megjelenése



3. ábra Egy fogazott hullámforma által generált frekvencia értékeket leíró hang megjelenése

Az ábrákon jól láthatók a hullámformák és a spektrumok közötti eltérések.

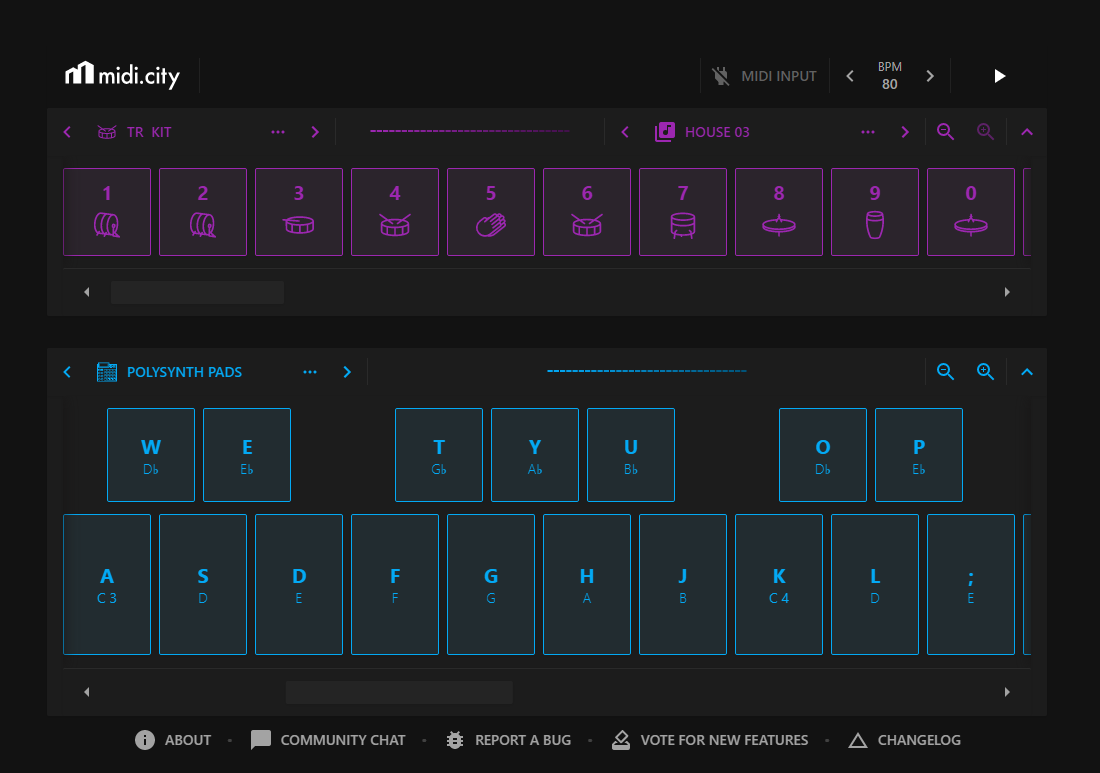
Mivel a ToneJS könyvtár ezen hangok generálást automatikusan elvégzi, nem szükséges a téma részletesebb érintése a dolgozatban.

# Versenytárs-elemzés

Természetesen léteznek hasonló szoftverek a piacon és ezeknek a szoftvereknek szeretném bemutatni mind az előnyeit, mind a hátrányait. Két lényeges terméket szeretnék bemutatni, amiknél kisseb elemzést végeztem.

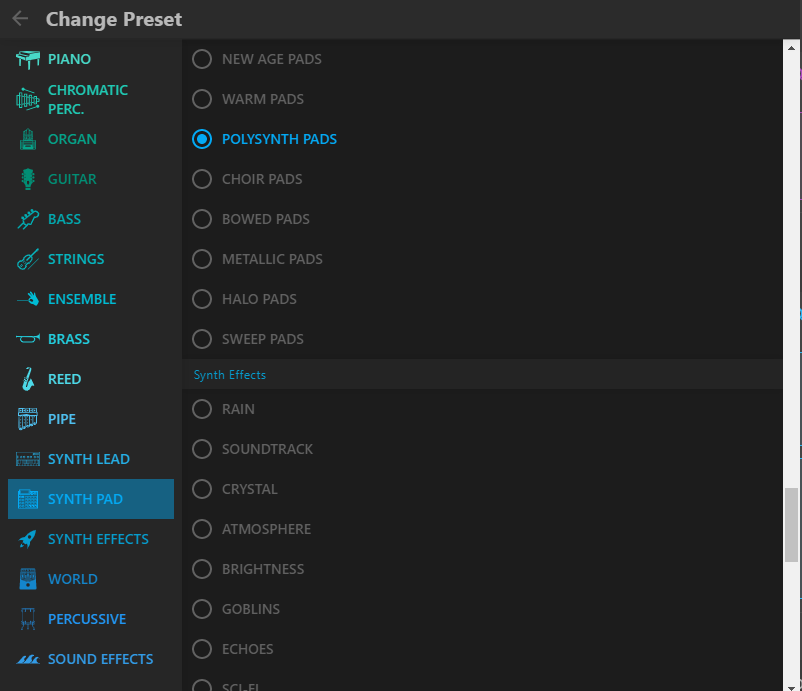
## MIDI.CITY

A midi.city egy webes felületen működő szintetizátor és dobgép. Ez egy tökéletes reprezentációja egy interaktív zenekészítésre alkalmas gépnek. Számos előre létrehozott hangbeállítást tartalmaz, amiket rövid pillanatokon belül ki is próbálhatunk. Használata nagyon egyszerű és egyértelmű. A szoftver bárki számára elérhető, ingyenes és szórakoztató, csak egy web böngészőre van szükség a használatához. Működtetését megtehetjük a billentyűzettel, vagy a képernyőn az ikonok rákattintásával is.



4. ábra A midi.city megjelenése a web böngészőben

A rengeteg beépített hangnak köszönhetően akár több órán keresztül is zenélhetünk és szórakozhatunk ezen a felületen



5. ábra A midi.city szoftverban megtalálható számos hangszer, és hangbeállítás

Látható, hogy mennyi féle hangszer és hangvariáció áll rendelkezésre, legyen az gitár, rézfúvós vagy emberi hang. Ezekre kattintva be is tölti nekünk az alkalmazás a hangot, ezzel lehetővé téve a felhasználó számára a megszólaltatását.

***Előnyei***

* Ingyenes
* Könnyen használható
* Rengeteg hangszer
* Dobgép

***Hátrányai***

* Nem személyre szabható hangok
* Kevés funkció

## Xfer Serum / SerumFX

A Serum egy zenei állomáshoz készített VST (Virtual Studio Technology) szoftver, aminek a segítségével összetett és modern hangokat készíthetünk kellő hozzá értés fejében. Ez a szoftver magában nem képes működni, szükséges hozzá egy DAW (Digital Audio Workstation) használata. Összetettségének köszönhetően, számos hasznos funkcióval rendelkezik, amit egy jó hang mérnök ki is tud használni. Nem csak meglévő hangok lejátszására képes, hanem többféle hanggenerálási módszert is el tud látni. Újonnan ez a legelterjedtebb szintetizátor a zenekészítők számára. Mivel egy ilyen összetett és hasznos szoftverről beszélünk, nem meglepetés, hogy az ára kiemelkedően magas. Emellett megemlítendő, hogy önmagában mit sem ér, mivel szüksége van egy zenei állomásra is a működéshez, aminek az ára ismételten nagyon magas.

Főbb funkciói közé sorolandó a hullámforma oszcillátor, aminek köszönhetően képesek vagyunk egyedi hangokat létrehozni. Akár képek feldolgozásával és felhasználásával is létrehozhatunk szintetizátorokat. Ezt társítva a szintén a Serum-ban megtalálható FX (hangeffektek) funkcióival nagyon hangzatos eredményeket érhetünk el.



6. ábra Az Xfer Serum megjelenése egy zenei állomáson belül



7. ábra Az Xfer SerumFX beállításainak reprezentációja

A képek alapján látható a két szintetizátor közötti különbség, funkcionalitás tekintetében.

***Előnyei***

* Viszonylag egyszerű használat a többi VST-hez képest
* Rengeteg hangképzési lehetőség
* Több automatizációra használható LFO és Envelope
* Hangeffektek
* Hullámforma személyre szabhatósága

***Hátrányai***

* Drága
* Telepítés
* DAW függő

Célom lenne, hogy az elkészített szoftver mind a két említett szintetizátor pozitív tulajdonságait át tudja emelni, és ezáltal létre lehessen hozni, egy ingyenes és könnyen használható zenekészítésre alkalmas környezetet.

# Technológiák

A fejlesztési technológiák kiválasztási folyamata közben számos szempontot vettem figyelembe, az adott feladat, és az elvárások függvényében a legalkalmasabb technológiák kiválasztására törekedtem.

A projektre vonatkozóan, a dokumentálhatóság, a költséghatékonyság, a biztos működés, a tartósság és tovább fejleszthetőségi lehetőségek egytől egyig olyan feltételek melyeket befolyásolt az alkalmazott technológia.

## Webfejlesztés

Webfejlesztés alatt egy olyan programozási módszertant értünk, ami két részből áll össze, egy front-end-ből és egy back-end-ből. Ennek a kettő rétegnek a kapcsolata hozza létre azt a programot, amit egy csapat, vagy egy fejlesztő készített.

***HTML***

Egy olyan leíró nyelv, melynek a használatával, weboldalak elkészítésére vagyunk képesek, és mára egy internetes szabvánnyá nőte ki magát. Aktuális változata a HTML5.

***CSS***

Egy stílust definiáló és leíró nyelv, aminek a segítségével, személyre szabhatjuk a weboldalalak kinézetét. Strukturált HTML típusú fájlok megjelenését határozza meg, és ezen felül XML-ek stílusának a leírását is megadhatja.

***JavaScript***

A JavaScript programozási nyelv, amely lehetővé teszi a bonyolultabb dolgok végrehajtását a weboldalakon. Minden olyan oldalon, ahol a weboldalon nem csak statikus információk vannak, hanem időszerű tartalmi frissítések is. Ez az általános webes technológiák harmadik rétege (HTML, CSS és JS).

Mivel ez a programozási nyelv hatalmas felhasználói bázissal rendelkezik, nagyon egyszerűvé teszi a tanulását. Bármilyen probléma is lépjen fel, az esetek nagy részében könnyen, és gyorsan megtalálhatjuk rá a választ az interneten. Emellett támogatja a már jól megszokott (*C#, Java stb*.) objektív orientált programozást is.

***jQuery***

Nagy népszerűségnek örvendő JavaScript nyelvhez készített könyvtár, amely a kapcsolatot a HTML és a kliens felületek között, egyszerű megoldásaival hozza lehetővé. Manapság már egyre kevésbé elterjedt, mivel a JavaScript már fel tudja váltani azokat a műveleteket, amik eddig csak a jQuery segítségével voltak lehetőek.

## Front-end

Egy rendszer a felhasználóhoz vissza juttatott kliens rétege. Feladata a felhasználó által kívánt műveletek elvégzése, és tájékoztatása a műveletek végrehajtásának kimenetéről. Célszerű egy letisztult, egyszerű UI létrehozása, hogy a végfelhasználó ne ugorjon olyan problémákba, amik rontják a szoftver használatának élményét. Ezt a műveletet érdemes párhuzamosan csinálni a UX design-al (User Experience Design) ami a felhasználói élmény lehető legjobb minőségére törekszik.

Számos front-end fejlesztésre alkalmas könyvtár létezik, amivel a Node szintén jól együtt tud működni, ilyenek például számos más nagy hírnévvel rendelkező keretrendszerek is, mint például a React vagy a Vue.

A front és back-end alapból függetlennek tekinthetők egymástól, de egy webes applikáció működtetéséhez nélkülözhetetlenek.

A keretrendszereket olyan módon készítették, hogy használatuk ne jelentsen problémát az átlagos fejlesztő számára. Habár többnyire eltérő gondolkodást igényelnek, így is egy programozói alaptudás szükséges hozzá, amit csak olyan módon tudunk elsajátítani, ha már egy egyszerűbbnek mondható keretrendszerrel töltöttünk el időt, vagy fejlesztettünk a segítségével. Több év tapasztalattal rendelkező fejlesztő mérnököknek is néha kihívást okozhat egy-egy újabb keretrendszer elsajátítása, mivel mindegyik minimálisan más gondolkodásmódot igényel, amire az emberi agynak rá kell állnia, tehát kulcsfontosságú a gyakorlat szerepe.

Nem tartottam fontosnak a független kezelését egymástól a front és back-endnek, mivel a fejlesztés során egyedül dolgoztam a szoftveren, ezért én nem használtam fel az említett keretrendszerek közül mindegyiket. Ehelyett egy egyszerűen értelmezhető szerveroldali sablonozó könyvtárat használok, aminek segítségével egyszerű módon lehet HTML fájlokat elkészíteni, amik mellett dinamikus tud adatokat kezelni.

Ilyen nyelvekről beszélünk a Handlebars, az EJS és a Pug esetében. Amelyikkel a legletisztultabb HTML fájlokat lehet készíteni, az az EJS, mivel felépítése teljesen megegyezik a megszokott HTML struktúrával.

### EJS

Az EJS (Embedded JavaScript) egy egyszerű sablonozó nyelv, amivel könnyen generálhatunk HTML kódot JavaScript segítségével.

Mivel JavaScript nyelvet használunk webfejlesztésnél, ezért nincs szükség megtanulni új iterációkat a használatlához, tehát gyorsan tudunk fejleszteni. Szintaxisa egyszerű, ahol a HTML-ben valamilyen műveletet szeretnénk elvégezni, csak körbe kell ölelni a JavaScript kódrészlet sorát scriplet jelölőkkel ( <% %>). [5]

<ul>

<% users.forEach(function(user){ %>

<%- include('user/show', {user: user}); %>

<% }); %>

</ul>

Látható, hogy a a jól megszokott HTML kódunkba is bele tudunk nyúlni egyszerű JavaScript kóddal. Ezeket csak egyszerűen körbe kell ölelni az EJS által előre definiált tag-ekkel.

Amikor a „ejs” fájlokat fordítja a compiler, build hibát fog dobni, ha hibásan írjuk a tag-eket a kódunkba. Ez egyben hibamegelőzési eszközként is szolgál, így kevesebb hiba fog kijutni valójában a front-end felületre, mivel a fejlesztő hamar észre tudja venni a hibát.

Használata ezen kívül teljesen megegyezik a megszokott HTML, CSS, és JS hármassal. Hasonló módon hívhatóak meg a JS fájlok, egy <script> tag-ben megadva forrásként, vagy a HTML fájlban kettő <script> tag közé zárva. Ha pedig egy stíluskinézetet szeretnénk csatlakoztati, azt is a megszokott módon tehetjük.

Egyetlen különbsége, hogy az elérési útnak a szerver applikációban, egy „views” nevű mappába keresi a forrásnevet, amit a gyökérmappában lehet megtalálni (ezt nem hozza létre automatikusan).

Legnagyobb előnye a fejlesztés során a kompatibilitásnak, a bootstrap támogatottsága.

### Bootstrap

A **Bootstrap** keretrendszer segít a webhelyek gyorsabb és könnyebb megtervezésében. Tartalmaz **HTML** és **CSS** alapú tervezősablonokat űrlapokhoz, gombokhoz, táblákhoz, navigációhoz.

Használata kettészeli a webfejlesztők közösségét, mivel bizonyos esetekben jobb, mint egy másik keretrendszer és fordítva. Jelenlegi verziója a Boostrap 4, ami számos változással járt az előző verzióhoz képest, és jelenleg is fejlesztés alatt van a Bootsrap 5. Mivel egy keretrendszerről beszélünk, ezért már más fejlesztők által előre megírt kódrészleteket hívjuk segítségünkre. Ezeket a HTML kódban lehet meghívni, egy CSS, és egy JS fájlt. Miután a HTML eléri ezeket a fájlokat, már használható is a keretrendszer. [6,7]

HTML-ben amikor elkészül a weboldal váza, elkezdünk foglalkozni a kinézetével. Ezeket megírhatjuk mi magunk sima CSS-ben, vagy használjuk a már beimportált „bootsrap.css” fájlunkat, amiből, ha szeretnénk valamit hozzá adni a HTML elemünkhöz, csak be kell írni az osztályai közé. A boostrap.js csak bizonyos funkciókhoz szükséges, mint például navigációs sáv és hasonlók.

Legnagyobb előnye, amiért mai napig előszeretettel használják, a rácsrendszer. Egy 12 hasábos, folyékony reszponzív rácsrendszer. Több törésponttal rendelkezik, 768 pixelnél, 992 pixelnél, 1200 pixelnél. Felépítésénél három szerkezeti egységnek van jelentősége, „.container” osztály jelölővel ellátott elem, ami alapból egy autómata margóval rendelkezik, ezáltal középre igazított felületről van szó. Ha szeretnénk teljes kép részére alkalmazni ezt a felületet ezt a „.container-fluid” osztállyal oldhatjuk meg. A konténeren belül sor osztálynevezővel ellátott elemek vannak, amiben további oszlop osztályú elemek vannak. Ezeknek az oszlopoknak lehet megadni az értékét, hogy mekkora részét foglalják el a sornak.

Emellett rengeteg más hasznos funkcióját használhatjuk, léteznek segédosztályok, űrlapok, gombok, képek, segédosztályok, komponensek.

### NexusUI

A NexusUI egy HTML5 és JavaScript-ből álló könyvtár. Funkcióival segít a böngészőnkben web audio hangszerek elkészítéséhez.

Mivel ez csak egy felületet ad időzítés és hangolás metódusokhoz, ezért önállóan hang generálására nem képes, viszont a Tone könyvtárral összekötve tökéletes működést biztosít. [8]

***HTML***

<div id="power"><div>

<div id="gain"><div>

***JS***

// Create interfaces

var power = new Nexus.Toggle("#power");

var gain = new Nexus.Slider("#gain");

Létrehozások az alábbi példákkal szemléltetve, nagyon egyszerű, mégis hatalmas segítséget nyújtanak különböző audio orientált szoftverek fejlesztése közben.

Nem csak hasznos, de jól is néz ki. Számos beállításra ad lehetőséget nekünk még a NexusUI, mivel azon kívül, hogy pár paraméter megadásával méretre szabhatjuk, színét változtathatjuk, ezek mellett egy nagyon komplex felépítésű objektum modellről beszélünk, aminek a minimum vagy maximum értékét is megadhatjuk, aminek a jelenlegi értékét, vagy változó értéket, egyből alkalmazhatjuk például frekvenciamodulációra, ezzel létrehozva egy analóg audio szintetizátorhoz hasonló működést. Gördülékeny hangzás eléréséért érdemes az értékek lépteit minél alacsonyabbra venni.

Ezesetben egyénileg kell beállítanunk a kívánt értékeket.

***Egyéni beállításokkal***

var dial = new Nexus.Dial('#target',{

'size': [75,75],

'interaction': 'radial', // "radial", "vertical", or "horizontal"

'mode': 'relative', // "absolute" or "relative"

'min': 0,

'max': 1,

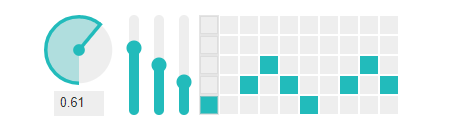
'step': 0.001,

'value': 0

})

Lehetőséget ad emellett, hogy az elemek értékét egyből szinkronizálja egy szám mezővel, aminek a megírása szintén majdnem egy sorban megírható.

A dolgozatban a legtöbbet használt eleme a „Dial”, mivel igyekeztem megtartani a szintetizátorok hagyományos beállító felületét megtervezni a szoftverben.



8. ábra NexusUI által szolgáltatott gombok, elemek

Ezen felül rengeteg más opciónk van, például egyszerű gombok használata, szám érték mező, csúszkák, választó gombok, zongorabillentyűk, sequencer, oszcilloszkóp, és még rengeteg más is.

Mivel alapból ez a könyvtár egy gyors, prototípusozó szoftvernek készült a felhasználó zenei ötleteinek sablonozására, vannak hiányai, mint például, hogy reszponzív tulajdonságokkal nem rendelkezik, így megnehezítve a kinézet személyre szabhatóságát.

## Back-end

A back-end felületén a rendszer azon rétegét értjük, ami a tényleges adatfeldogozási műveletekért felelős. Feladata a kliensoldalról érkező információk fogadása és feldolgozása, és a már feldolgozott eredmény visszaküldése a kliens oldalra.

### Node.js

A **Node** egy **JavaScript** környezet, amely a Google Chrome által használt V8 motorra alapul. Olyan tulajdonságokkal rendelkezik, ami alkalmas közepes szinten álló webes alkalmazások fejlesztésére.

Sokkal erősebb teljesítményt mutat, a nem blokkoló eseményvezérelt modellnek köszönhetően, így előnyt jelent a szál alapú szerver környezetekkel szemben (PHP, Ruby, Rails), amik az I/O-t blokkolják, mivel egyidőben több felhasználót kezel, külön szálak rotációjával. A Node így folyamatosabb futást, több alkalmazás párhuzamosítását, valamint az egész rendszer optimálisabb működését teszi lehetővé. [9]

***Jellemzői***

* Gyors működés.
* Esemény vezérelt.
* Jól használható Networking.
* Jól használható API szolgáltatások.
* Rengeteg könyvtár.
* Kritikus alkalmazások futtatására is alkalmas.
* Könnyen tanulható.

A node package manager-en belül a csomagrendezőben (*NPM*) elérhető további számos bővítmény. Telepítésük egyszerűen, az *npm install* paranncsal hajtható végre, ami után a választott csomag címét írjuk. Ha rögzíteni szeretnénk a telepítendő bővítményt, akkor a *--save* kulcsszó hozzáadásával a parancs végére, így bekerül az applikációra nézve függőségi kapcsolatként az ezeket tároló fájlba. Miután a töltés befejeződött, a bővítménycsomag megjelenik a node által létrehozott modulok mappában.

#### Könyvtárak

Megeshet, hogy egy bizonyos könyvtár felhasználásához szükség van más könyvtárakra is, ezért előre definiálva van az, hogy milyen egyéb könyvtárak hozatalára van még igény.

Az express, egy webalapú keretrendszer, amiben minden olyan webapplikáció beüzemeléséhez szükséges parancsot és funckiót megtalálunk, ami szükséges lehet. Az express számos másik könyvtártól is függ, ezért előre meghatározva kell rögzíteni, hogy melyik verzió használatával volt problémamentes működés. Ezen információk megtekinthetőek a node által létrehozott JSON fájlba, amibe automatikusan íródik bele ez telepítéskor.

#### Konfiguráció

A Node alkalmazás létrehozásakor, néhány informatív jellegű adatot (Szerző, Leírás, licensz, verzió stb.) állíthatunk be, ami szintén a Node által létrehozott JSON fájlba kerül eltárolásra. Különféle scripteket is beállíthatunk, hogy bizonyos kulcsszavakra indítsuk el a szervert, élesben vagy akár fejlesztésben.

#### Futtatás

A fájl, amit a „*node*” terminál paranccsal lefuttatunk (leggyakrabban „*app.js*”), ezt követően elindul egy „localhost” címen futó lokális szerver, amit a fájlban megadott porton érünk el.

### Express

Az express alatt egy olyan gyors és minimalista webkeretrendszer értünk a Node.js-en belül, ami egy olyan technológia, mint egy köztes szoftver, ami segíti az applikációnak a szerver működtetését és a route-ok irányítását. Más szóval, a Node is függ az Express-től, hogy egy zökkenőmentes működést tudjon így biztosítani.

Az API (Application Programming Interface) lehetővé teszi a felhasználóknak, a front-end és az adatbázis közötti beállítások, és útvonalak megadását a beérkező és a kimenő kérésekhez.

Előnyei közé tartozik, hogy támogatja a webes alkalmazások szerveroldali MVC (Model-Nézet-Vezérlő) architektúráját. Emellet kiemelendő, hogy számos többi csomagot is támogat, mint például a már említett sablonozó könyvtárakat is (Pug, EJS, Ruby, stb). Ezek mindegyike személyre szabhatóak, legyen akár szó route-okról, vagy bővítményekről.

Az Express.js rövid időn belül megmutatta, hogy hatalmas népszerűségre tehet szert, mivel az egyszerűen használható módjainak és funkcióinak köszönhetően, a mai napig az egyik legkeresettebb és legsikeresebb Node.js könyvtár. Több mint 48 000 csillaggal rendelkezik a jelen pillanatban is a GitHub felületen.

Kijelenthető, hogy a keretrendszer, funkcionalitását és lehetőségeit tekintve, a jövőben is nagy szerepet fog betölteni a webfejlesztők számára.

### Nodemon

A Nodemon egy fejlesztést segítő könyvtár. Figyeli a változásokat a projekt fájljaiban, változás esetén újraindítja a szervert.

## Web technológiai alapok

Ezt a webtechnológiai rövid ismertetőt az Express szabványaira írom, amely elengedhetetlen az alkalmazásban használt logika megértéshez. Azon főbb részeket kiválasztva, melyek előtérbe vannak helyezve a fejlesztésben.

### Routing

A „route” -ok (elérhetőségek), alatt az alkalmazásban azokat a parancsokat értjük, ami alapján reagál a kérésekre egy végponton. Ez egy elérési út, és egy http kérés eggyüttes működése.

Ezek az útvonalak rendelkezhetnek akár több funkcióval is, amik az elérésekor kerülnek végrehajtásra. Ilyenkor egy elérési kérés megy ki a szerver oldalra.

A következő struktúra alapján hozhatóak ezek létre. [10,11]

Ahol:

* app: a node szerver fájljában meghívott express parancs
* METHOD: http metódus kérésre
* PATH: Egy út
* HANDLER: A függvény, amit meghívunk

Get paranccsal a következő:

var express = require(’express’)

var app = express()

app.get(’/’, (req,res) => {

res.send(’hello world’)

})

### Metódusai

A http-ből származó metódusok hozzákapcsolhatók az express példányához, így megalkotva a parancsot. Nem csak az express képes ezeknek a parancsoknak a végrehajtására.

GET

Fájlok vagy adatok kéréseinek kiszolgálására.

POST

Fájlok, vagy adatok kiküldése a szervernek. Új elemek is létrehozhatóak a metódussal.

PUT

Egy meglévő adatok felül írunk.

PATCH

Csak egy bizonyos részét szeretnénk frissíteni.

DELETE

Adatok törlésére.

### Státuszkód

A státuszkódok extra információként működnek, amit a böngészőnek ad át, és ennek az információnak a tudatában segít eldönteni, hogy egy parancs vagy folyamat sikerült-e vagy nem. Minden státuszkód más hibát mutat, ezekből a legismertebb a 404, ami annyit jelent, hogy a keresett elem vagy információ nem található vagy nem létezik. [12]

1. táblázat Státuszkódok táblázata, és magyarázata

|  |  |  |
| --- | --- | --- |
| 2xx | 200 | Sikeres művelet |
|  | 201 | Sikeres egység létrehozása |
| 3xx | 301 | Áthelyezve |
| 4xx | 401 | Nincs autentikálva |
|  | 403 | Nincs hozzáférése |
|  | 404 | Nem létezik |
|  | 422 | Hibás input |
| 5xx | 500 | Szerver hiba |

### Middleware

Ilyen esetben egy olyan függvényről beszélünk, ami átveszi a kérés, és a válasz objektumokat, amiket szerkeszthetünk, bővíthetünk, majd miután végeztünk vele továbbítjuk a következő függvénynek felhasználva a „next()” parancsot.

Az egyszálúság, ami annyit jelent, hogy a felhasználók megosztják ugyanezt a memóriaterületet a Node-on belül. Nem ajánlott olyan adatokat tárolni, amik felhasználóspecifikusak lehetnek, mivel ezek más kapcsolatokon keresztül is könnyedén elérhetők. [11]

#### Body-parser

A body-parser egy olyan köztes szoftver, ami arra szolgál, hogy a bejövő kéréseket feldolgozható formába csomagolja ki. Röviden, kinyeri az adatot a request header-ökből, amit JSON fájlokhoz hasonlóan tud később feldolgozni.

## Tone.js

A Tone.js egy olyan keretrendszer JavaScripth-hez, aminek a segítségével interaktív módon, zenét készíthetünk a böngészőnkben. Fejlett ütemezési lehetőségeket, hangszintézist, hang effekteket, és intuitív zenei absztrakciókat kínál és biztosít a Web Audio API tetejére. [13]

Mivel rengeteg funkcióval rendelkezik, ezért egy nagyméretű keretrendszerről beszélünk, de emellett hasonlóan erős is. Számos metódusának köszönhetően, bármi zeneileg elképzelhetőt létrehozhatunk vele.

Olyannyira összetett, hogy audio mérnököknek is lehetőséget nyújt használatra. A keretrendszer weboldalán részletes dokumentációt találunk róla, arról, hogy milyen egyéb lehetőségeket nyújt még a hanggeneráláson felül, megnézhetjük a közösség által „tone” segítségével létrehozott fejlesztéseket is. Teljes mértékben nyílt forráskódú, tehát használata is ingyenes. Az elérhető Demo alkalmazásokat tekintve jól látható a sokrétűsége, mivel különböző felhasználási területeit próbálhatjuk itt ki.

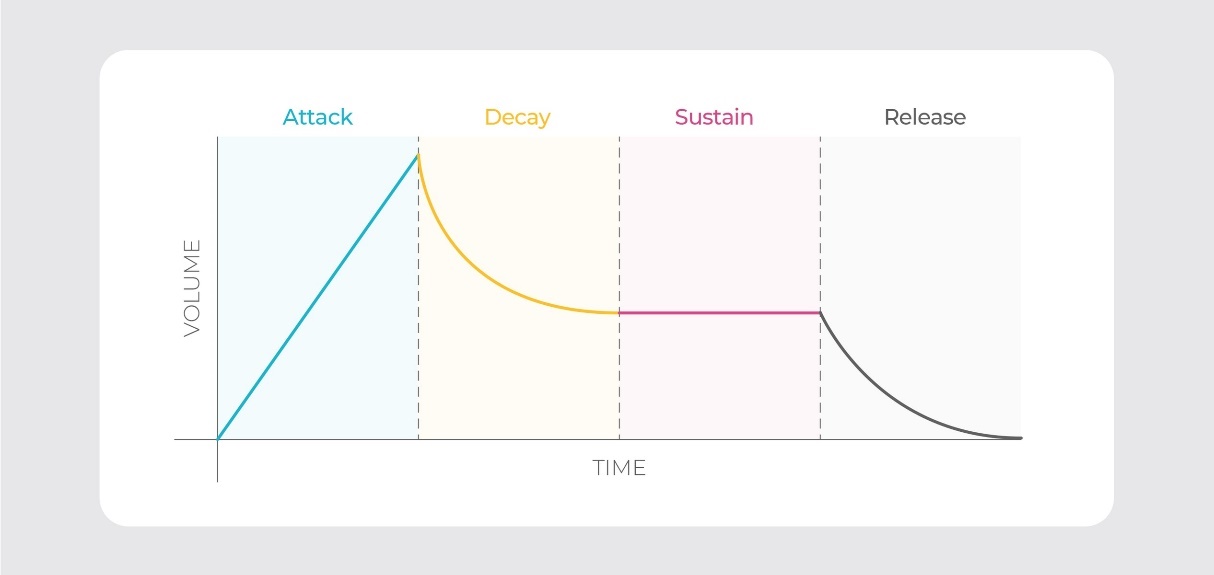
***Moduláris***

Néhányan igyekeztek egy olyan alkalmazás kifejlesztésében, ami nagy részben hasonlít egy moduláris szintetizátor felépítéséhez. Ezeknek a használta viszont nem a legegyszerűbb, mivel előzetes tapasztalatot igényel. Mivel itt nem mindegy milyen sorrendben kötjük össze a modulokat, továbbá az sem mindegy, hogy ezeket a kötéseket melyik adapterből melyik adapterbe kötjük. Ha hibázunk egyáltalán nem fog nekünk megszólaltatni akármilyen hangot, vagy csak szimplán egy nem kívánt eredményt kapunk. Elegendő idő eltöltésével viszont jó szórakozást nyújthat audio mérnököknek, vagy csak egy felhasználónak, aki hobbiként tekint a témára, vagy egyszerűen érdekli a szintetizátorok működése.

***További megoldások***

A további demókat böngészve, kiderül mégis mi másra képes még a keretrendszer. Számos szoftver az én dolgozatomhoz hasonlóan egy olyan felületet hozott létre, ahol interaktív módon lehet kipróbálni a „Tone” által nyújtott funkciókat. Ezeket letisztult módon prezentálja, és egyszerű a használatuk, de ez negatív tulajdonságokkal is jár. Az oldal semmi másra nem képes, mindegyik demo csak egy feladatot lát el, nincs egységesítve. Vegyük példának a „Ramsophone” demóját, ez csak egy előre megadott hangsorozatot játszik vissza, aminek tudjuk állítani az ADSR (Attack, Delay, Sustain, Release) beállításait, ezek a hang amplitudóját modulálják.

Ennek ábrával való magyarázata a következő:



9. ábra Egy egyszerű ADSR envelope képi reprezentációja

**ATTACK:** ez határozza meg hogy mennyi idő alatt jut el a hang amplitudója a legmagasabb értékre

**DECAY:** amikor időben elérkezett az ATTACK maximális értékéhez, az amplitudó vissza csökken a DECAY által leírt görbe értékei szerint.

**SUSTAIN:** ezt az állapotot egészen addig tartja fenn a hang, amíg szól, vagy le van nyomja egy zongorabillentyű

**RELEASE:** ennek a görbének az értékei akkor lépnek érvénybe, amikor megszűnik a hang generálása, például miután felengedtük a zongorabillentyűt

Ez szemlélteti, hogy mégis milyen erős keretrendszert értünk a Tone alatt. Hasonló működést is létre tudunk hozni az LFO-val (Low-Frequency-Oscillation), ami csak részben tér el az ENVELOPE-tól. A főbb különbség, hogy ez folyamatosan hajtódik végre, míg az envelope csak egyszer, amikor elindul a hang. Ezeknek az LFO-knak beállítható több hullámforma is, mint például szinusz, négyzetes, háromszög, fogazott, vagy véletlenszerű.

Ezeket használva, megközelíthető egy modern szintetizátor működésének kivitelezése, amik licenszének az ára megközelítőleg 60-70 ezer forint, és itt egy példányról beszélünk.

Fejlett ütemezési funkcióinak köszönhetően dobgépek létrehozására is alkalmas, és ha ezeket összekötjük egy több sávot játszó lejátszóhoz, létrehozható egy komplett zenei állomás a böngészőnkben.

***Hangszerek***

Legegyszerűbben az alábbi módon hozható létre egy hangszer:

//szintetizátor létrehozása és hozzákötése a fő kimenethez (hangszoró)

var synth = new Tone.Synth().toMaster();

//Középső C hang lejátszása, egy nyolcad hosszúságú ideig

synth.triggerAttackRelease('C4', '8n');

Itt a generált hangot, hozzákötjük egy Tone beépülő modulhoz, ami egy egyszerű Attack-Release szűrőn engedi át a nyers hangot mielőtt az a fő kimenetre csatlakozna.

***Effektek***

Képesek együtt működni a létrehozott hangszerekkel. Hangeffektjeinket is teljesen személyre szabhatjuk paramétermegadással. Ezek valós időben is változtathatóak, tehát ha szól egy hang, annak frekvenciaértéket aközben is változtathatjuk miközben az szól, és mindezt megszakítás nélkül végzi, ami egy kisimult és tiszta élményt nyújt. Ezek a szabályok egyaránt érvényesek az effekteknél, ahol minden modulnak a saját értéket változtathatjuk, legyen szó frekvenciáról, wet/dry beállításról, állapotról (általában ki és be kapcsolásra). Teljes mértékben effekt függők ezek a paraméterek, mivel amíg egy torzító (distortion) effektnek beállítható csak a torzítás erőssége, addig egy automatikus filterező-nek (auto-filter) más beállításai is vannak, mint LFO frekvencia, hullámforma értékek, intenzitás és még rengeteg más.

Az alábbi módon hozható létre hangeffekt, és ennek a hozzákötése a hangszerünkhöz:

var distortion = new Tone.Distortion(0.6) //effekt deklarálása 0.6

var tremolo = new Tone.Tremolo().start() //deklarálása alap értékkel

// 4 hangú polifónikus szintetizátor deklarálása

// chain metódus segítségével több effektet is hozzá tudunk csatolni

// először vezeti át a generált hangot egy torzítón, utána egy tremolo

// effekten, és utána küldi csak ki a hangszórókra

var polySynth = new Tone.PolySynth(4, Tone.Synth).chain(distortion, tremolo, Tone.Master)

// egérgomb lenyomására meghívódó akkord lejátszása

document.querySelector('#chord').addEventListener('mousedown', () => {

polySynth.triggerAttack(['C4', 'E4', 'G4', 'B4'])

})

// egérgomb felengedésére meghívódó akkord leállítása

document.querySelector('#chord').addEventListener('mouseup', () => {

polySynth.triggerRelease(['C4', 'E4', 'G4', 'B4'])

})

Ezek a legfőképp kiemelendő tulajdonságok a Tone keretrendszerről.

## MongoDB

A MongoDB szintén egy nyílt forráskódú adatbázis szoftver, ami a NoSQL adatbázisszerverek közé tartozik.

Használatát azért találom jelentősen fontosnak, mivel nagyszerűen illeszkedik több webfejlesztési „stack”-be is, mint például a MERN (MongoDB, Express, React, Node), vagy a MEAN (MongoDB, Express, Angular, Node). Ezek elterjedt fejlesztéstechnológiai kupacok, amit a webfejlesztők előszeretettel alkalmaznak, egészében, vagy részlegesen, egy új kupacot kialakítva. Teljes mértékben egyén függő a választás, de természetesen minden egyes „stack”-nek vannak előnyei, és hátrányai is.

Az adatokat JSON-hoz hasonló formátumban tárolja, ezáltal egyszerű megoldást nyújt az adatok eléréséhez, módisításához, vagy törléséhez. Számos nagyobb felhasználó is ezt az adatbázisrendszert használja a mai napig, mint például az MTV Networks, Craiglist, FourSquare, és még sokan mások.

Ennek az adatbázisnak a használata, több különböző NPM-el telepíthető könyvtárat igényel, ezzel biztosítva folyamatos, és biztonságos működést a felhasználó, és a fejlesztő számára.

### Mongod

A mongod egy MongoDB által használt háttér folyamat. Főbb jelentősége, hogy a szerveroldali feladatokat könnyedén kezelni tudja, például beérkező kérések fogadása, kliensoldali felhasználó értesítése, memória menedzsment. A „mongo” egy parancs, ami kapcsolatban van a klienssel (például, rendszer adminisztrátorok és fejlesztők).

### Mongoose

A mongoose szintén egy MongoDB-hez szükséges objektumadat modellező segédkönyvtár a Node-on belül, aminek a segítségével kezelhetjük az adatok közötti kapcsolatokat, és szolgáltat a fejlesztőknek séma validációkat.

### Passport

***passport***

A Passport egy autentikációs middleware a Node-on belül. Meglehetősen rugalmas és moduláris, mivel túlzás nélkül, bármiféle olyan webalkalmazásba egyszerűen be lehet építeni, aminek a szervere express-t használ.

Többféle beléptető lehetőséget kínál, ezek a stratégiák közé tartozik a jól megszokott felhasználónév, és jelszó kombinációja, vagy akár Facebook-os, és Twitter-es autentikációt is be lehet építeni vele az alkalmazásunkba, nagyobb erőfeszítés nélkül. Ezeket a stratégiákat mind külön kell telepíteni az npm-en keresztül.

***passport*-local**

Beléptetési stratégia, ami a lokálisan tárolt felhasználónevet és jelszót leellenőrizve autentikál. Ennek a gördülékeny kommunikációját az adatbázissal a passport-local-mongoose könyvtár teszi lehetővé. Az itt eltárolt jelszavakat kódolva (hashelve) teszi meg, így nyújtva egy minimális biztonságot a végfelhasználók számára.

# Célkitűzés

A bevezető fejezetben említettem, hogy a zenekészítés elkezdéséhez egy bizonyos háttér tudásra van szükség. Célom, hogy egy fundamentális tudást át tudjak adni a felhasználónak, és a későbbiekben, egy bizonyos alaptudással rendelkezzen, amit már alkalmazni is tud egy zenei produkciós szoftverben (DAW – Digital Audio Workstation). Mivel az ilyen szoftvereknek ára meglehetősen magas, ezért aki érdekelt a témában, de még nincsen semmilyen előzetes tapasztalata, nem biztos, hogy szívesen költene ennek a használatára. A szoftver használata ingyenes és bárki számára elérhető, aki rendelkezik egy web-böngészővel ellátott számítástechnikai eszközzel. Továbbá célom, hogy miután a felhasználó a weboldalon kipróbálja, és megérti a használatát az audio szintézisnek és a hang effekteknek, élesben is tudja gyakorolni. Két lényeges felület segítené ezt, egy szintetizátor felület, és egy tanuló felület.

***Tanuló felület***

A tanuló felületen megtalálhatóak a szintetizátorban használt effektek és oszcillátorok, ezekhez társul egy rövid lényegre törő magyarázat, ami tartalmaz effekthasználathoz való tanácsokat, és továbbá megtudhatja a felhasználó működésének is a hátterét.

***Szintetizátor felület***

A szintetizátor felületen rendelkezésre áll egy, vagy több oszcillátor (hang generálására szolgál), ezen generált hangokat egy zongora billentyűihez hasonló gombok működtetnek (összesen 36 billentyű). Ezeket a billentyűket aktiválhatjuk egérkattintással, vagy a billentyűzeten gomblenyomással. A generált hangra rá lehet kötni hang effekteket, és ezeknek az értékét (wet/dry ennek az értéke adja meg mennyi részben vezeti át a kimenő hangot az effekten) is lehet módosítani. Miután létrejött a kívánt hang, van mentésre lehetőség, aminek a segítségével bármikor újra használhatjuk az elmentet szintetizátor beállításokat.

***Landing-felület***

Az oldal betöltése után ide irányít minket a szoftver, itt egy rövid leírást lehet elolvasni magáról a programról, funkcióiról, és arról, hogy miért jött lére. Bootstrap keretrendszer segítségével lehetőség nyílik reszponzív oldal létrehozására, tehát mind telefonon és számítógépen is gördülékenyen használható.

***Gyűjtemény felület***

A gyűjtemény felületen, megtalálhatóak az elmentett szintetizátor beállítások. Ezekre rákattintva a szoftver visszajátszik egy előre megadott dallamot, azokkal a hang beállításokkal, amiket a felhasználó beállított amikor készítette. Ezek az értékek egy adatbázisból hívhatóak elő. Felhasználók a bejelentkezés után ide menthetik a beállításaikat, amit bármelyik másik felhasználó is felhasználhat, és megjegyzéseket is hagyhat alattuk építő kritika jelleggel.

***Bejelentkező felület***

A felhasználónak biztosított felület, hogy a már előlegesen regisztrált felhasználójába be tudjon lépni.

***Regisztrációs felület***

A felhasználónak biztosított felület, hogy regisztrálni tudjon további funkciók eléréséhez.

# Megvalósítás

A szoftver megvalósítása folyamán, igyekeztem létrehozni egy egyszerű és igazán letisztult szoftvert, mivel a felhasználói élmény írja körül az egész szoftver működését és megbízhatóságát. A szoftver webes megvalósítása mellett azért döntöttem, mert szerettem volna webfejlesztési gyakorlatot is szerezni a szoftver elkészítése alatt, ami személyes véleményem szerint sikerült is. Emellett hasznosnak is találom a webes megoldást, mivel ennek a technológiának az elterjedtségének köszönhetően, bárki számára könnyen elérhetővé válik, mivel manapság már majdnem minden személy rendelkezik egy olyan számitástechnikai eszközzel, ami képes egy web böngésző futtatására. A szoftver megvalósítása során elkészítettem a célul kitűzött felületeket, apróbb módosításokkal, javításokkal.

* Kezdő oldal
* Közösségi felület / mentett elemek felülete.
* Autentikációra alkalmas felületek.
* Szintetizátor felülete.
* Tanulásra alkalmas, hasznos információkkal ellátott felület.

## Szerver

A weboldal egy Node szerver segítségével fut az alkalmazás hátterében. Indítását a „node server.js” paranccsal tudjuk elvégezni, aminek következtében máris elérthetővé válik a projekt tartalma. Miután lefutott a parancsunk a terminálban logolva látható, hogy az adatbázis „seed” fájlja is sikeresen lefutott, ami arra szolgál, hogy konzisztens maradjon az adatbázis, ezért a lokális felületünkre is legenerálja számunkra a szükséges adatokat. A szerver fájlban „localhost” -os címen van beállítva, és ehhez van egy konstans változóban definiálva a port száma, ami a 8000-es. Miután felállt a rendszer, bármelyik preferált web böngészőben megnyitható az oldal, a „localhost:8000” URL-en. A szoftver alapból az alapútra, route (’/’) -ra vezérel minket, ami a kezdő oldalt jelenti ezesetben.

## Kezdő oldal

Az oldal betöltésekor különféle információkat találhatunk a szoftverről, mint hogy melyik célközönséget célozza meg, milyen funkciókat lát el, és milyen felületekkel rendelkezik.

A design-t a bootstrap rács rendszere segíti, aminek köszönhetően reszponzívvá válik az oldal. Reszponzív oldal alatt azt értjük, hogy az oldalon szereplő elemek alkalmazkodnak a mérethez, tehát egy telefonos és egy mobilos megjelenés között kisebb különbségeket lehet felfedezni.



10. ábra Egy asztali számítógép monitorján megjelenő kezdő oldal, aminek a design-ját a bootstrap rács rendszere segíti.

A rácsrendszer szerint mindig 12 részre van felosztva egy elemeket tartalmazó konténer. Ha az alábbi képet nézzük látható, hogy itt a sorban két 6 értéket felvevő oszlopról beszélünk, így felezhető le a rácsnak a sora. A fele-fele elrendezésnek köszönhetően sikerült bele építeni pár minimalista dizájn elemet is, aminek a segítségével egy esztétikus kinézetet sikerül nyújtani a felhasználónak.

A lap tetején található a navigációs menü, ami ismét a bootstrap keretrendszer része. Itt találhatóak a linkek, amire kattintva elérjük a szoftver többi route-ját, oldalát.



11. ábra Ugyan az a kezdő oldal megjelenése egy mobil eszközön, ami a reszponzív tulajdonságoknak köszönhetően működőképesen, és jól jelenik meg.

A telefonos nézet az alábbi alapján jelenik meg a felhasználónak, ahol az eddig elosztott oszlopok, vagy teljesen eltűnésre vannak beállítva, vagy egyszerűen egymás alá pakolódnak. A képek osztályainak is megadható reszponzív tulajdonság, amik hasonló módon átméreteződnek, így jó megjelenést biztosítva.

Az eddig egy sorban található navigációs menü összeomlik, és egy gomb jelenik meg helyette, amire kattintva előkerülnek az eltűntetett menü linkek.

A weboldal alján még továbbá megtalálhatóak az oldal elérhetőségi linkjei a Facebook, Soundcloud, GitHub, és LinkedIn felületein.

## Autentikáció

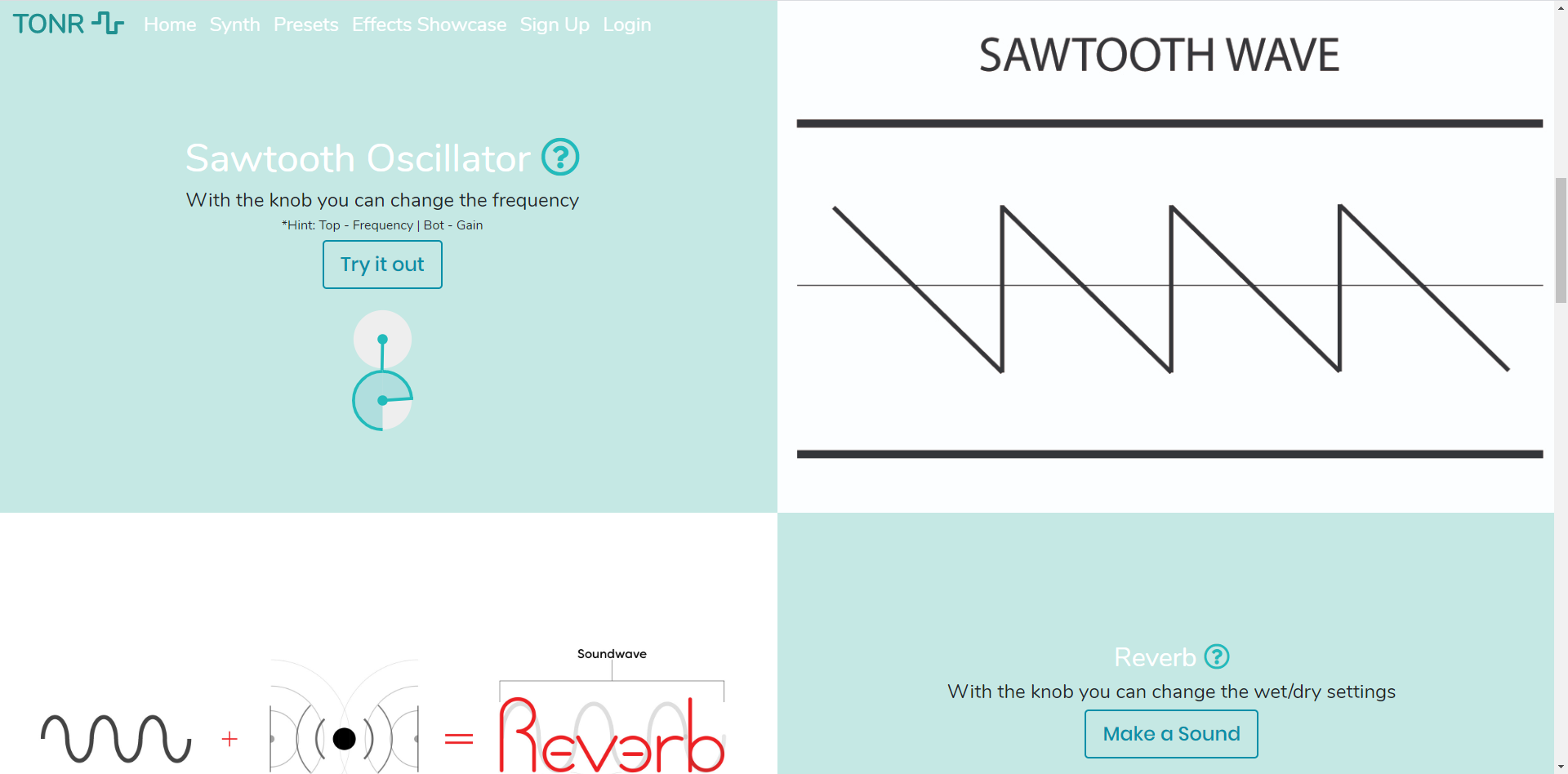
Az autentikációs feladatot két külön oldal látja el, egy bejelentkező és egy regisztrációs felület. A regisztrációs oldalon két input mező található, egy a felhasználónévnek, és egy a jelszónak. Az adatok beírása után a leadás gombra kattintva ezt az adatot elküldi az adatbázisnak, ahol ez tárolásra kerül, és a jelszavak hash-elt formában ezzel nyújtva egy minimális védelmet a felhasználónak. Ezután egyből beléptet minket az új fiókkal. Legközelebbi belépés folyamán már használhatjuk ezt a felhasználónkat.

A jelenlegi belépett felhasználó nevét a szoftver a belépést követően beilleszti a navigációs menübe, emellett eltűntetni a belépés és regisztrációs gombokat, és a helyére egy kilépés gomb kerül.

## Tanuló felület

Ezen a felületen megtalálhatóak a szintetizátorban használt elemek külön elválasztva egymástól. Ezek mindegyikéhez tartozik egy gomb, ami megszólaltat egy hangot, és alatta NexusUI könyvtárból használt tekerő gombok, amik az adott elem, hangeffekt beállításait módosítja. Ennek köszönhetően a felhasználó interaktív módon kipróbálhatja a zenekészítésben is használt effekteket. Ezekhez társul még egy kérdőjel ikon is, amire kattintva tovább információk jeleníthetőek meg, mint például effektek működésének magyarázata, használati tanácsok és angol nyelvű magyarázó videók, egy „bootstrap modal” felugró ablaknak köszönhetően.

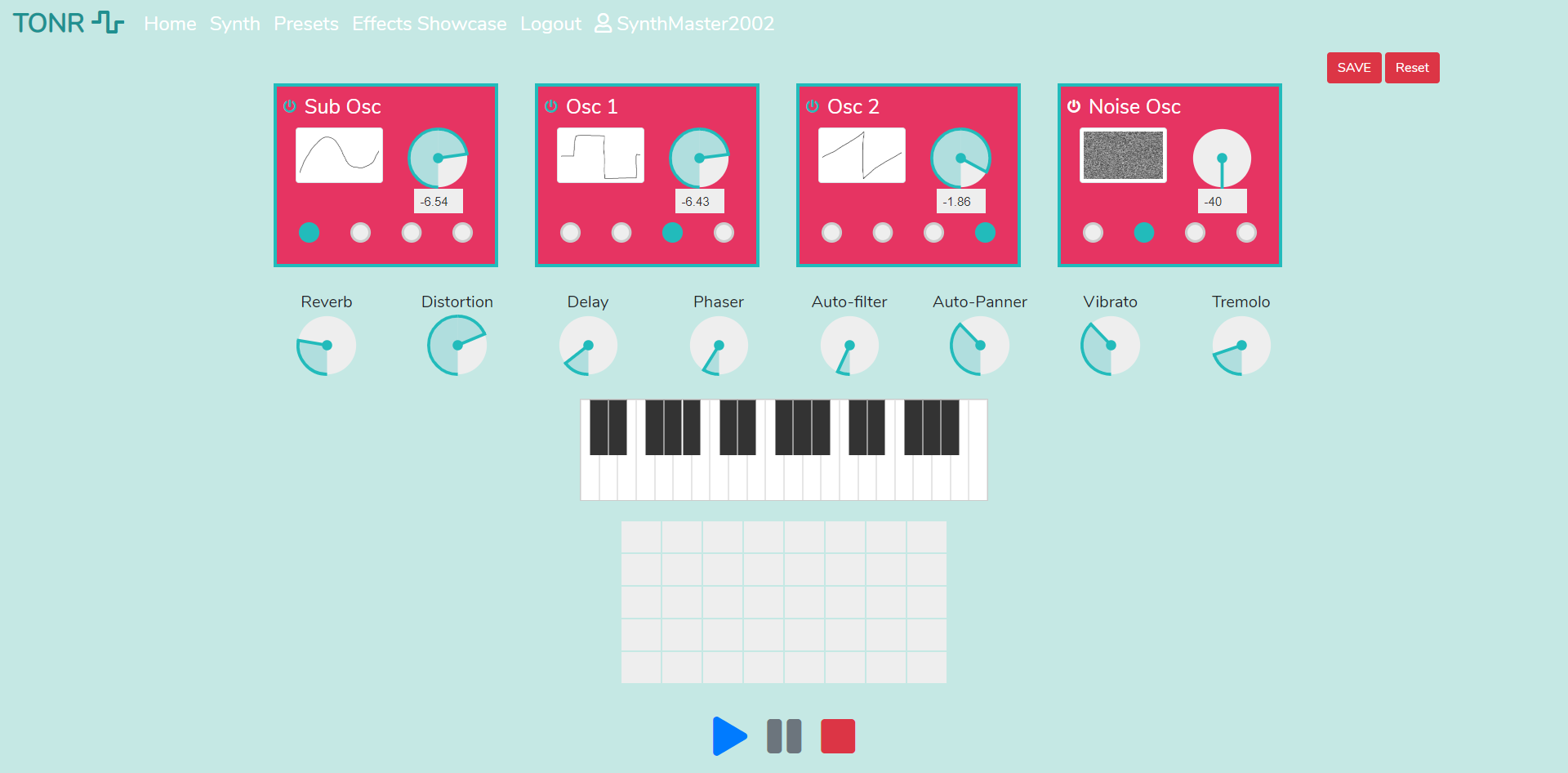
A kiválasztott elem könnyű elérését egy lap tetején található gomb menü segíti, amikre kattintva az oldal azon részére navigál minket egy animáció segítségével, ahol az található.



12. ábra A tanuló felület effektes szektciója ahol interaktív módon kipróbálhatóak a hangeffektek

## Szintetizátor

A szintetizátor felületen, elérhető a felhasználó számára egy egyszerű 4 oszcillátor által működtetett hangeszköz. Minden hanggenerálóhoz tartozik egy ki és be kapcsoló gomb, egy amplitudó érték módosító tekerő, és egy hullámforma választó menüsor. Egy kép megjelentésével jól látható, hogy éppen az adott oszcillátoron milyen hullámforma van beállítva, emellett a jelenlegi amplitudó értéket decibelben is lehet látni. A szoftverben mind a 4 generáló definiálva van a Tone JS segítségével és az ehhez tartozó értékeket a NexusUI által szolgáltatott beépített elemekkel, és gombokkal lehet módosítani. Kikapcsolt helyzetben működésképtelen helyzetben van az oszcillátor, tehát nem fog hangot megszólaltatni egészen addig, amíg be nem kapcsoljuk. Lehet egyszerre négyet is használni, de természetesen, ha csak egyet, kettőt, vagy hármat szeretnénk egy időben az is lehetséges.

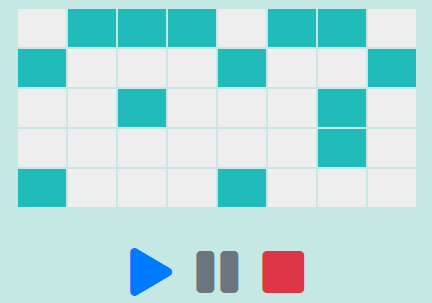


13. ábra A szintetizátor felület kinézete

Az elemek elrendezését itt is a bootstrap rács hálózata segíti, így könnyen stílusos megjelenést ad a felületnek. A hanggeneráló sor alatt látható egy tekerő elemekből álló hangeffekt sor is címezve. Ezek mindegyike szintúgy definiálva van a kódban alap értékbeállításokkal, amit a ToneJS állít be, kivétel a effekt intenzitás beállítás, amit a tekerők segítségével állíthat be a felhasználó kedve szerint. Alap helyzetben csak az egyes, és a kettes számú oszcillátorra van néhány rákötve, mivel mélyfrekvenciákon tegyük fel a visszhang effektus erősen rontja a generált hang minőségét. Miután beállítottuk az értékeket, és erősségeiket, az effekt sor alatt látható egy zongorával megegyező 36 billentyűs gomb elrendezés, amit szintén a NexusUI szolgáltat. Az elem kattintásra, és gomb lenyomására is aktiválható, úgy, hogy a billentyűzeten gomb lent tartásra hívódnak meg a hanggenerálókat megszólaltató függvények. Jelenleg monofónikus módban üzemel csak a szoftver, tehát ha egyszerre több gombok nyom le a felhasználó, mindig a legutoljára lenyomott billentyű fogja meghívni a függvényt, tehát nem képes akkordok egy idejű lejátszására. Gomb felengedésére egy másik függvény hívódik meg, ami az oszcillátor leállását biztosítja. Ez a függvény egészen addig nem hívódik meg, amíg a 36 billentyű közül mindegyik állapot értéke nem hamis, tehát egyik billentyű sincs lenyomva, így szolgáltatva egy kellemes és gördülékeny zenei élményt.

Mivel a számítógép billentyűzete nehezen fedi le a 36 billentyűt úgy, hogy annak használata még hasonlítson egy zongoráéhoz, így csak 12 hangot lehet megszólaltatni vele, ami egy egész oktáv és a fél-hangok. Ha mind a 36 gombot szeretnénk aktiválni, akkor léptető gombok segítségével, ami fel, és leléptet a szintetizátor oktávjain, máris elérhető lesz.

A felület még kiegészül egy dob géppel, ami különböző dob hangeffektusokat játszik vissza, ami alapbeállítás szerint 120 ütemet játszik vissza egy perc alatt. Használata meglehetősen egyszerű, csak rákattintunk azokra a négyzetekre, amik szeretnénk, hogy hangot vissza játszanak egy ismétlődés alatt.



14. ábra Szintetizátor felülethez tartozó dobgép lejátszó, szüneteltető, és újraindító gombokkal

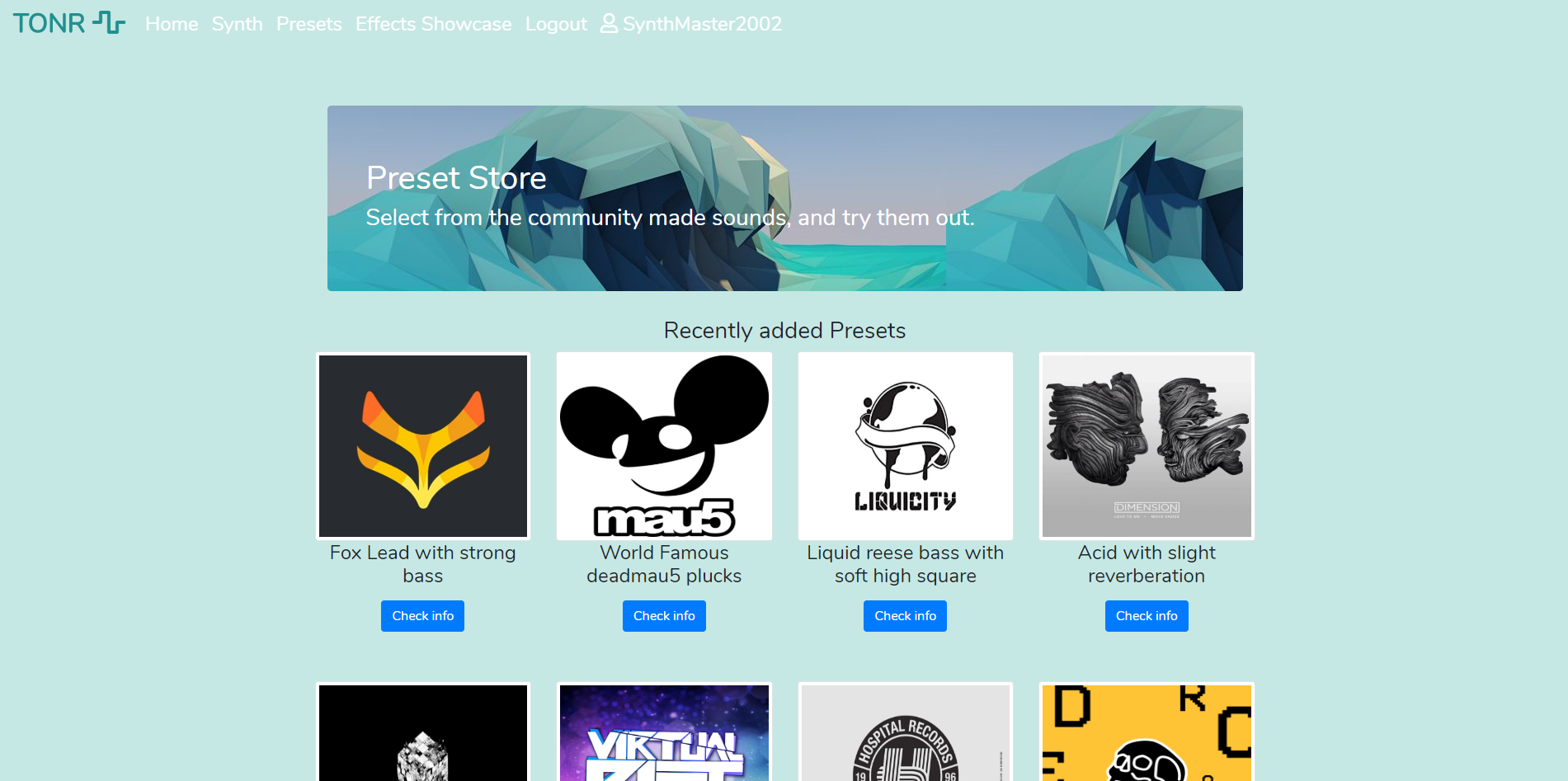
A lejátszás gomba kattintva visszajátssza a berajzolt szekvenciát, ami egy mélydob, rövid/hosszú cin, taps, és tam, ezt folyamatosan visszajátssza, amíg nem szüneteltetjük, vagy leállítjuk. Szüneteltetés esetében onnan tudja folytatni az ütemet, ahol megállt, leállításnál az elejéről kezdi.

A felhasználó miután létrehozta a beállításokat, amiket preferál, van lehetősége menteni ezeket. A lap tetején található mentés gombra kattintva jelenít meg, egy bootstrap modal-t, ahol két beviteli mező található, egy a beállítás nevének, egy pedig egy kép URL-jének. Miután a felhasználó rákattint a leadás gombra, a szoftver elmenti az összes háttérben futó szintetizátorhoz kapcsolódó értéket egy adatbázisba, a hozzá megadott névvel, és képpel. Ezek később a mentett elemek felületén lesz megtalálható.

Amennyiben valami olyan beállításokat sikerült megadni a felhasználónak, amivel nincs megelégedve, a visszaállítás gomb segítségével beállíthatóak az oldal betöltésekor is alkalmazott alap értékek. A rendszer figyelmeztet ilyenkor, hogy az eddigi munka mind törlésre kerül, és csak a felhasználó megerősítésével történik meg a visszaállítás.

## Mentett elemek felülete

A mentett felületen megtalálhatóak az összes felhasználó által mentett hangbeállítások. Listázásnál az adatbázis objektumain végighaladva jeleníti meg az elemeket, egy címmel és egy képpel. Mindegyik elemhez tartozik egy gomb, amire kattintva több információ jeleníthető meg a hangbeállításról.



15. ábra Elemek kilistázott nézete a mentett elemek felületén

Az alkalmazás ilyenkor az objektum ID-ja alapján vezérel minket egy külön oldalra, ahol az objektumban tárolt hangbeállításokat is el lehet érni. Ugyan úgy jeleníti meg a képet, és a címet, amihez társul 3 külön gomb. A lejátszás gombra kattintva egy előre megírt dallamot játszik vissza a szoftver, azokkal a hangbeállításokkal, amik hozzá tartoznak. Alatta található gombra kattintva lekéri ezeket az adatokat, és átvezérel minket a szintetizátor felületre, ahova betölti a már lekért paraméter értékeket, így felhasználható már saját hangra, ez a hangbeállítás. Ezekhez az elemekhez hagyhat a felhasználó saját kommentet, amivel kifejezheti a véleményét a beállításokkal kapcsolatban, kritizálhatja, vagy megdicsérheti, és akár tanácsokat is adhat a szerző számára. Ilyenkor egy másik oldalra navigál az oldal, ahol felismeri a belépett felhasználót, így csak azzal a felhasználónévvel hagyhat kommentet. Amennyiben nincs belépve egy regisztrált fiók sem, átvezérel a bejelentkező felületre, ahol be tud lépni a felhasználó, hogy kommentet írhasson.

# Továbbfejlesztési lehetőségek

Mivel a szoftver rendelkezik már alapvető és hasznos funkcióival, lehet ötletelgetni, hogy még hányféle módon lehetne bővíteni a weboldalt, hogy a használata még inkább hasonlítson egy zenei állomáséhoz, aminek megvalósításával más célközönségeket is be lehet vonni a használatba. Idő hiányában az alábbi funkciók kifejlesztésére nem jutott sor, de a továbbiakban tudásomhoz mérve tervezem bővíteni az alkalmazást.

## Hangsávok

Amíg a szoftver jelenlegi helyzetben is rendelkezik már két hangsávval, de ezeknek a hozzáférhetősége igencsak limitált. Számos hangsávval szeretném bővíteni az alkalmazást, ahova a létrehozott dallamainkat, és dob mintáinkat helyezhetjük el. Segítségével akár kezdetleges dalokat és készíthetünk már a böngészőnkben.

## Projekt megosztás

A már a hangsávos verzióban létrehozott készítményeket, a Gyűjteményfelületen hasonló módon meg lehetne osztani, mint az előző verzióban létrehozott hangokat, dob alapokat.

## Kollaboráció

Felhasználói rendszer kiépítése után, kettő vagy akár több felhasználó is közösen neki kezdhet, vagy módosíthat egy projektet. Közös munkájukat ugyan olyan módon megoszthatnák a többi felhasználóval. Ez a módszer jól megszokott a zenei állomásoknál, de ott minden esetben egy projekt állományt küldenek el valamilyen platform segítségével. Célom, hogy egy valós idejű felületet hozzak létre, amihez minden kollaboráló hozzá fér egy időben.

## Automatizáció

Ez is egy jól megszokott funkciója a zenei állomásoknak, ennek a segítségével tudjuk automatizálni az értékeket. Vegyünk példának mondjuk egy egyszerű alacsonyáteresztő filtert. Ha ennek a levágási értékét egy hangsáv idejének szinkronizált görbéjével automatizáljuk, nagyon hangzatos eredményeket kaphatunk. Ezt az automatizációt olyan módon szeretném létrehozni, hogy a létrehozott automatizációs görbét rá lehessen kötni bármelyik értékre. Ha a hangerőt szeretnénk, vagy a frekvenciát, vagy bármilyen hangeffektnek az erősségét változtatni, arra mind tudjon megoldást nyújtani a funkció. Ennek használata már megszokott a zenei állomásoknál, ezért tartom elengedhetetlennek implementálását a szoftverbe.

## Dob variációk

Jelenleg a sequencer beépített 4-5 hanggal rendelkezik. Ez elegendő, hogy egy alap dob mintát létrehozzunk, amire tudunk játszani saját dallamot, szeretném ezt kibővíteni olyan módon, hogy a felhasználó lokális mp3 fájlok segítségével, vagy egy legördülő menüben kiválasztva is meg tudja szólaltatni a betöltött hangot. Kibővíteni a sorok számát, hogy 20 féle hangot is meglehessen szólaltatni egy loop alatt.

## Hangszerek

Mivel egy szintetizátor kifejlesztése is huzamosabb ideig tartott, nem maradt idő több hangszer implementálására. A ToneJS számos más hangszer elkészítésére is ad lehetőséget, mint például egy Sampler (hangfájlt módosít a billentyűkhöz), vagy Granular (hangfájlt módosít a billentyűkhöz számos beállítással). Ezen felül különböző oscillator típusok közül is választhatunk, mint például AM, FM PWM szintetizátor.

## MIDI

Egy alap MIDI támogatottságot is szeretnék még beépíteni a szoftverbe, aminek a segítségével más programokban készített dallamokat is betölthetnénk. Ehhez társulna még a MIDI kontroller támogatottsága, amihez a Tone ismét segítséget nyújt. MIDI feltöltése a szoftverbe és felismerése, később MIDI exportálása a szoftverből.

## Exportálás

A kész projekteket MP3, vagy WAV formátumban lenne lehetőségünk kiexportálni, amit már majdnem minden elektronikus eszköz fel tud dolgozni. Emellett MIDI exportálás lehetőségének is az implementálása.

# Összefoglalás

Egy zenekészítésre alkalmas szoftver kifejlesztése, ami kiegészül egy alapvető közösségi, és tanuló felülettel, nem egyszerű feladat. Bár különböző technológiák megismerése után, és ezek használatával kivitelezhetővé vált a megalkotása. Az alkalmazás fejlesztése során sikerült gyakorlatot szereznem webalapú technológiákban, így már magabiztosabban fogok tudni neki állni a következő projektnek.

Jelen dolgozatomban ismertettem a digitális analóg átalakítók működését, a hangszintézis alapjait, és a legfontosabb versenytársakat a jelenlegi piacon. Emellett részletes magyarázatot írtam a felhasznált technológiákról, amiből kiderülhet, miért az adott eszközre, könyvtárra esett a választás.

A dolgozat célja, hogy azokat a felhasználókat segítse, és szórakoztassa, akik még nem rendelkeznek semmilyen zenei hardverrel, szoftverrel, vagy tudással, de szeretnének valamilyen módon neki vágni ennek az elfoglaltságnak. Ennek a célnak köszönhetően jött létre az alkalmazás, ahol a felhasználó ezeket a fundamentális információkat megszerezheti, és elkezdhet egyszerű módon zenét készíteni. Interaktív felületeinek köszönhetően számos lehetősége van megismerni a felhasználónak a zenekészítésben is használt hangeffekteket, azok hasznosságát, és felhasználási tanácsait. Közösségi platform létének köszönhetően, más felhasználók is adhatnak tanácsokat egymásnak, kifejezhetik véleményeiket, megoszthatják saját munkáikat, így kivitelezve a további tanulási lehetőségeket.

A jövőben terveim közé tartozik az alkalmazás továbbfejlesztése, melynek során az alkalmazás még inkább átalakul egy zenei állomássá, ami az alapvető zenei produkciós funkciókat tartalmazza. Több hangszer implementálására kerülne sor, számos hangsáv párhuzamos visszajátszására, automatizációs beállítás érték módosítására, kollaborációs lehetőségekre, projekt megosztásra, hang exportálására, a dobgép teljes átalakítására és alapvető MIDI olvasás, és írás lehetőségeire. Továbbá optimálisabbá, gyorsabbá, és tisztábbá refaktorizálni a forráskódját a szoftvernek, így biztosítva egy letisztultabb működést. Alkalmazni később szoftvertervezésben használt szerkezeti mintákat is, mint például az MVC (Modell-Nézet-Vezérlő) is, amit használva egyértelműbbé és egyszerűbbé válhat a fejlesztés folyamata.

.

# *Irodalomjegyzék*

[1] Dr. Lipovszki György (2012): Jelfeldolgozás és számítógépes irányítás -  
<https://regi.tankonyvtar.hu/hu/tartalom/tamop412A/2010-0017_36_jelfeldolgozas_es_szamitogepes_iranyitas/adatok.html>

[2] Végh Andás: A jel definíciója -

<https://veghandras.webnode.hu/news/a-jel-definicioja-is-jelekkel-tortenik-/>

[3] Dr. Fodor Dénes: Digitális jelfeldolgozás (2014 - Pannon Egyetem) -

<https://regi.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0042_digitalis_jelfeldolgozas/adatok.html>

[4] BME: Digitális szűrés -

<http://www.mogi.bme.hu/TAMOP/mereselmelet/ch10.html>

[5] EJS hivatalos weboldala -

<https://ejs.co/>

[6] Ádám Laki: Bootstrap bemutató -

<https://adamlaki.com/hu/bootstrap-bemutato/>

[7] Bootstrap hivatalos dokumentáció -

<https://getbootstrap.com/docs/>

[8] NexusUI hivatalos dokumentáció -

<https://nexus-js.github.io/ui/api/>

[9] Node -

<https://medium.com/javascript-scene/introduction-to-node-express-90c431f9e6fd>

[10] Express -

<https://expressjs.com/en/guide/routing.html>

[11] Express Middleware usage -

<https://expressjs.com/en/guide/using-middleware.html>

[12] HTTP státusz kódok -

<https://httpstatuses.com/>

[13] ToneJS hivatalos dokumentáció -

<https://tonejs.github.io/docs/>

# *Ábrák jegyzéke*

[1. ábra Egy digitális analóg jelátalakító ( DSP ) egyszerű reprezentációja 4](#_Toc38808587)

[2. ábra Egy háromszög hullámforma által generált frekvencia értékeket leíró hang megjelenése 8](#_Toc38808588)

[3. ábra Egy fogazott hullámforma által generált frekvencia értékeket leíró hang megjelenése 9](#_Toc38808589)

[4. ábra A midi.city megjelenése a web böngészőben 10](#_Toc38808590)

[5. ábra A midi.city szoftverban megtalálható számos hangszer, és hangbeállítás 11](#_Toc38808591)

[6. ábra Az Xfer Serum megjelenése egy zenei állomáson belül 12](#_Toc38808592)

[7. ábra Az Xfer SerumFX beállításainak reprezentációja 13](#_Toc38808593)

[8. ábra NexusUI által szolgáltatott gombok, elemek 19](#_Toc38808594)

[9. ábra Egy egyszerű ADSR envelope képi reprezentációja 26](#_Toc38808595)

[10. ábra Egy asztali számítógép monitorján megjelenő kezdő oldal, aminek a design-ját a bootstrap rács rendszere segíti. 34](#_Toc38808596)

[11. ábra Ugyan az a kezdő oldal megjelenése egy mobil eszközön, ami a reszponzív tulajdonságoknak köszönhetően működőképesen, és jól jelenik meg. 35](#_Toc38808597)

[12. ábra A tanuló felület effektes szektciója ahol interaktív módon kipróbálhatóak a hangeffektek 36](#_Toc38808598)

[13. ábra A szintetizátor felület kinézete 37](#_Toc38808599)

[14. ábra Szintetizátor felülethez tartozó dobgép lejátszó, szüneteltető, és újraindító gombokkal 38](#_Toc38808600)

[15. ábra Elemek kilistázott nézete a mentett elemek felületén 39](#_Toc38808601)

# *Táblázatok jegyzéke*

[1. táblázat Státuszkódok táblázata, és magyarázata 24](#_Toc39315992)

# *CD Melléklet*

* /
  + dolgozat/
    - Szakdolgozat\_CSM\_Tervezes\_2.docx
    - Szakdolgozat\_CSM\_Tervezes\_2.pdf
  + szoftver/
    - node\_modules
      * …
    - public
      * css
      * imgs
      * js
      * models
    - views
      * pages
      * partials
    - package.json
    - package-lock.json
    - seeds.js
    - server.js