Sintetitzador sono-visual per a navegadors web

Memòria del Treball de Final de Grau

Grau de Multimèdia

Desenvolupament d'aplicacions interactives

Autor: Pere Amengual Gomila

Consultor: Kenneth Capseta Nieto

Professor: Carlos Casado Martinez

14 de Gener de 2018

Crèdits/Copyright

Dedicatòria

Dedicat a totes les persones que decideixen compartir el seu coneixement amb la resta del món.

Cita

"The mathematician's patterns, like the painter's or the poet's must be beautiful; the ideas, like the colors or the words must fit together in a harmonious way. Beauty is the first test: there is no permanent place in this world for ugly mathematics."

G. H. Hardy (1877 - 1947), A Mathematician's Apology, Cambridge University Press, 1994.

https://www.cut-the-knot.org/manifesto/beauty.shtml

Abstract

Disseny i implementació d'un sistema amb capacitat de generar sons i imatges en temps real inspirat en el funcionament de les consoles làser dels anys setanta. Aquest sistema permet la creació d'una gran varietat de sons de tipus experimental i la seva corresponent visualització en temps real seguint els principis de l'anomenada "música visual".

La principal tecnologia que fa possible les complexes tècniques de generació sonora és la API Web Audio (1), dissenyada per manipular i reproduir elements d'àudio en una pàgina web o una aplicació. La llibreria Tone.js permet configurar complexos enrutaments similars als dels sintetitzadors modulars, mentre que NexusUI ens ajuda a configurar una interfície gràfica d'usuari amb tots els elements típics dels instruments electrònics virtuals.

Entre les principals característiques d'aquest sistema destaquen, d'una banda, la utilització d'oscil·ladors de quadratura, els quals presenten dos senyals a la seva sortida amb una diferència de fase de 90 graus. Aquesta correlació de fase específica és determinant per la creació de patrons visuals atractius coneguts amb el nom de corbes de Lissajous. Aquests patrons bidimensionals es formen quan els valors de dos senyals amb relacions específiques de freqüència i fase són assignats a les coordenades dels eixos x i y d'una gràfica, respectivament.

Addicionalment, el sistema implementa una matriu de modulació, és a dir, un sistema que permet assignar un senyal de modulació arbitrari a qualsevol de les destinacions de modulació possibles. Amb aquest sistema, podem controlar de forma gràfica el flux dels senyals dins el sistema, així com crear de forma fàcil rutes de modulació complexes que poden incloure camins de realimentació.

Una altre característica destacable en aquest sistema és la implementació de preajustaments (*presets*) d'usuari, mitjançant els quals s'agrupen tots els paràmetres del sintetitzador sota un únic índex, que permet enregistrar l'estat del sistema en un moment determinat, emmagatzemar-lo en memòria i recuperar-lo posteriorment. Per implementar aquest sistema de preajustaments es fa servir MongoDB, una base de dades orientada a documents, formant part d'un *stack* que fa servir el grup de tecnologies MEAN (MongoDB, Express.js, AngularJS i Node.js).

Paraules clau

música visual, figures de Lissajous, consola làser, web audio, sintetitzador, oscil·ladors de quadratura, matriu de modulació.

Abstract (English version)

Design and implementation of a system able to generate sounds and images in real time inspired by the operation of the laser consoles of the seventies. This system allows the creation of a large variety of experimental sounds and their corresponding visualization in real time following the principles of the so-called "visual music".

The main technology that makes possible the technical sound generation techniques is the Web Audio API (1), designed to manipulate and reproduce audio elements in a web page or application. The Tone.js library allows for the configuration of complex routings similar to those found in modular synthesizers, while NexusUI helps us configure a graphical user interface with all the typical elements of virtual electronic instruments.

Among the main features of this system, on the one hand, we find the use of quadrature oscillators, which present two signals at their output with a phase difference of 90 degrees. This specific phase correlation is decisive for the creation of attractive visual patterns known as the Lissajous figures. These two-dimensional patterns are formed when the values of two signals with specific relationships of frequency and phase are assigned to the coordinates of the x and y axes of a graph, respectively.

Additionally, the system implements a modulation matrix, a system that allows you to assign an arbitrary modulation signal to any of the possible modulation destinations. With this system, we can control graphically the flow of signals within the system, as well as easily create complex modulation paths that may include feedback paths.

Another remarkable characteristic in this system is the implementation of user *presets*, by means of which all the parameters of the synthesizer under a single index are grouped. These *presets* allow the recording of the system's state at any specific moment, by storing it in memory and retrieving it afterwards. To implement this preset System, a document-oriented database such as MongoDB is used as part of a stack that is defined by group of technologies know as MEAN (MongoDB, Express.js, AngularJS and Node.js).

Keywords

visual music, Lissajous figures, laser console, web audio, synthesizer, quadrature oscillators, modulation matrix.

Notacions i Convencions

Títol de capítol: Helvetica: negreta, 24pt, blau fosc, fons blau

Títol de secció: Helvetica, negreta, 13pt, negre

Títol de subsecció: Helvetica, negreta cursiva, 12pt, negre

Text normal: Helvetica, regular, 12pt, negre

Text en cursiva: Helvetica regular, 12, negre

Text ressaltat: Helvetica, negreta, 12pt, negre

Enllaç a internet: Helvetica, regular, 9pt, blau

Fragment de codi: Menlo, negreta, normal i cursiva (segons la sintaxi del codi)

```
qm.engine.busses = function () {
    qm.engine.reverb.receive("reverbChan");
    //************comentari *********
}
```

Text per notes a peu de pàgina: Helvetica, regular, 9pt, negre

> *Text de cita: Helvetica, cursiva, 12pt, blau*

Agraïments

A la meva família.

Índex

Figures i taules

Índex d'imatges

Índex de taules

1. Introducció

1.1 Motivacions personals

Des de fa més de trenta anys la creació sonora experimental ha estat la meva passió, i en alguns moments, també la meva dedicació professional. Encara que l'accés als mitjans de creació digital era molt restringit en els meus inicis, vaig fer tot el possible per integrar les tecnologies digitals en els meus treballs fins que, finalment, vaig decidir emprendre els estudis del grau de multimèdia amb la finalitat d'incorporar els nous mitjans al meu portfoli creatiu.

Dins el camp de la creació sono-visual, l'any 2014 vaig presentar al festival ME_MMIX la peça "Estudis sobre corbes Lissajous", una performance en la que un complex sintetitzador modular crea els patrons visuals i sonor que un làser projecta sobre una superfície tridimensional. També al mateix any vaig presentar a Berlin, conjuntament amb Luca Zerbinatti, la instal·lació interactiva "Thereministic" en la que el públic interactua amb uns sensors per crear senzilles però impactants patrons de llums projectats en temps real.

Dins els estudis del grau he descobert les enormes possibilitats que entorns de programació com Processing ofereixen als creadors sono-visual. A l'assignatura "Tractament i publicació d'àudio" vàrem descobrir la possibilitat de programar un sintetitzador de sons amb codi (fent servir Processing i la llibreria Minim) i més tard, a altres assignatures com "Disseny d'interacció" i "Realitat virtual" vaig tenir l'oportunitat de seguir investigant en la creació d'interfícies d'usuari destinades a la creació sonora i visual dins l'àmbit performatiu.

Arribada l'hora d'afrontar el Treball final de grau, he optat per desenvolupar una aplicació web completa (amb *front-end* i *back-end*) ja que he pensat que seria una bona ocasió per repassar les habilitats adquirides en el desenvolupament web i alhora aprofitar l'oportunitat per investigar en les possibilitats que ofereix la relativament recent API Web Audio, així com l'estructura de *back-end* MEAN (MongoDB, Express.js, Angular, Node.js)

1.2 Justificació del projecte

Un dels motius que fan preferible el desenvolupament d'un instrument sono-visual d'arquitectura fixa, com el del projecte que ens ocupa, sobre un d'arquitectura modular que es podria desenvolupar fent servir alguns dels entorns més coneguts dins l'àmbit de la creació sonora com Max, Pure Data o Supercollider, és la seva major facilitat d'ús i senzillesa conceptual, que fan possible una corba d'aprenentatge més ràpida i, per tant, millor adequació a les tasques de improvisació i interpretació performativa. Un instrument amb una estructura i paràmetres fixes és més fàcil d'aprendre a tocar i, per tant, a curt termini oferirà més bons resultats.

També seria possible desenvolupar una aplicació com la que es presenta en aquest treball fent servir plataformes diferents, com podria ser per exemple un microcontrolador de tipus Arduino com Teensy, que amb la seva llibreria d'àudio ens permetria també configuracions i resultats molt semblants als que pretenem aconseguir en el projecte que ens ocupa. Si s'ha escollit la Web i les possibilitats dels navegadors ha estat precisament per la universalitat i omnipresència d'aquests darrers, el que ens permet difondre una proposta estètica particular d'una forma pràcticament immediata i a un cost reduït. A més, els coneixements adquirits en el desenvolupament del projecte poden ser aplicats en altres àmbits com el de la creació artística a la xarxa: net-art, art interactiu, art generatiu, etc.

1.3 Antecedents històrics

a. Un antecedent directe de la proposta estètica que defineix el projecte es el que es coneix amb el nom de “música visual” (2), que fa ús d’estructures musicals dins les imatges visuals, així com mètodes que poden traduir sons o música en una presentació visual relacionada. Entre els autors més representatius d’aquesta disciplina destaquen, entre d’altres, Oscar Fischer i James Whitney.

b. En els anys setanta i vuitanta del segle passat, abans que la tecnologia digital es fes servir per controlar els projectors làser, es feien servir consoles làser analògiques els senyals elèctrics que controlen el desplaçament dels servo-motors d’aquests projectors s’obtenien mitjançant dispositius electrònics analògics molt semblants a un sintetitzador musical, però amb la finalitat específica de crear figures geomètriques, conegudes actualment com *abstracts*. Un exemple d’aquest tipus de consoles analògiques, que encara es pot adquirir actualment, és la Z5, *d’Aldebaraan Laser Systems* (3).

c. El mètode principal de creació sonora que es fa servir en el projecte es coneix amb el nom de síntesi FM (4), una tècnica de creació sonora que fa servir la modulació en freqüència d’un senyal anomenat “portadora” per part d’un altra senyal anomenat “moduladora”. La relació de freqüències entre aquests dos senyals i l’índex (o quantitat) de modulació defineixen l’espectre harmònic del senyal resultant.

d. El sistema de enrutament, que rep el nom de matriu de modulació (5), que es fa servir en projecte s’inspira en el que utilitzen els sintetitzadors analògics amb *patchbay.* Els *patchbay* són uns panells de connexions que permeten enrutar els senyals dins un sistema fent servir uns connectors anomenats “pins” o “agulles” que es punxen a la matriu. Aquesta matriu presenta totes les fonts de senyal com sortides disponibles a cada fila de connectors amb destinació a les entrades de modulació disponibles, situades a cada una de les columnes de la matriu. Així, inserint un pin en una posició determinada de la matriu s’envia el senyal de la fila corresponent cap a l’entrada de modulació que es troba assignada a la columna seleccionada. Un exemple clàssic d’aquest tipus de sintetitzadors és el Synthi A, de la casa EMS, que varen fer servir artistes com Pink Floyd o Brian Eno. Actualment, els sistemes modulars de Rob Hordijk (6) també fan servir aquest tipus de matrius, encara que amb una fiabilitat i característiques tècniques (*buffering*, inseriments, etc.) molt superiors.

2. Descripció

El resultat del present treball de final de grau serà una aplicació web anomenada **Quadramat**, en relació a dues de les seves dues característiques principals: l'ús d'oscil·ladors de quadratura i la matriu de modulació. Aquest sistema té la capacitat de generar sons i imatges en temps real de manera semblant a com ho feien les consoles làser dels anys setanta.

Els oscil·ladors de quadratura són el component encarregat de generar les oscil·lacions que posteriorment es convertiran en sons i imatges. Aquest tipus d'oscil·ladors presenten dos senyals a la seva sortida amb una diferència de fase de 90 graus i, precisament, aquesta correlació de fase específica és determinant per la creació de patrons visuals atractius.

Els canvis en el valor del senyal produïts pels oscil·ladors produeix les anteriorment esmentades corbes de Lissajous (7), uns patrons bidimensionals que es formen quan els valors de dos senyals amb relacions específiques de freqüència i fase són assignats a les coordenades dels eixos x i y d'una gràfica, respectivament. Per exemple, un senyal sinusoïdal i un cosinoïdal de la mateixa freqüència formen un cercle, mentre que si la relació de freqüències es 2:1 es forma una figura semblant a un vuit.

Però un sol oscil·lador no produeix patrons sonors o visuals massa atractius, i per aquest motiu és necessari connectar-los entre ells. Així, podem tant sumar el resultat de les seves oscil·lacions com fer que el valor d'un d'ells afecti de manera immediata algun valor dels altres oscil·ladors, o fins i tot d'ell mateix. Una manera eficient d'implementar aquestes connexions és el que es coneix com "matriu de modulació", un sistema que permet assignar un senyal de modulació arbitrari a qualsevol de les destinacions de modulació possibles. Addicionalment, algunes matrius de modulació permeten especificar la quantitat de modulació a enviar. Amb aquest sistema, podem controlar de forma gràfica el flux dels senyals dins el sistema, així com crear de forma fàcil rutes de modulació complexes que poden incloure camins de realimentació.

Amb la finalitat de poder recuperar fàcilment els valors del paràmetres que formen una figura sono-visual, s'ha implementat un sistema de preajustaments (*presets*) d'usuari, que permet l'agrupació de tots els paràmetres del sintetitzador sota un únic índex, enregistrant l'estat del sistema en un moment determinat, per posteriorment emmagatzemar-lo en memòria i permetre la seva recuperació posterior.

3. Objectius

3.1 Principals

L'aplicació web, en la seva versió final, ha d'implementar aquesta funcionalitat

- Creació de senyals sonors sobre dos canals correlacionats en fase
- Interfície de disseny sonor completa, amb possibilitat de modulació creuada de paràmetres com amplitud (bipolar) i freqüència (*thru-zero* FM)
- Visualització de les corbes de Lissajous a la finestra del navegador
- Emmagatzematge de preajustaments de tots els paràmetres *(presets)* en un servidor

3.2 Secundaris

Addicionalment, l'aplicació podria disposar de mòduls addicionals destinats a augmentar la seva funcionalitat

- Mòduls destinats a crear senyals de control de baixa freqüència (envolupant, seqüenciador...)
- Matriu ampliada per facilitar l'ús performatiu de l'instrument, seguint les directrius d'interfície d'usuari més adequades per un instrument electrònic d'aquestes característiques
- Component social i de compartició dels preajustaments per part dels usuaris

3.3 Principals reptes a assolir per la consecució dels objectius

- Elaboració d'un diagrama de blocs del sintetitzador i el seu prototipat en un entorn de disseny sonor d'alt nivell (Kyma, Clavia G2 o similar) per assegurar que es compleixen les expectatives respecte de les possibilitats sonores i visuals de l'aplicació a desenvolupar
- Estudi de les possibilitats de les llibreries de *front-end* (Tone.js i NexusUI) i posterior implementació del prototip en la plataforma escollida
- Estudi del funcionament de l'estructura de *back-end* (MEAN) i posta en marxa del servidor
- Creació del model de dades per l'emmagatzemat dels preajustaments *(presets)*

4. Escenari actual de les eines de creació sono-visual

En el moment de redactar aquest document, les tècniques de creació sono-visuals basades en entorns de programació són les que es fan servir més freqüentment, encara que també és cert que pot observar un cert ressorgiment de tècniques que fan servir exclusivament maquinari i, més concretament, dispositius analògics inspirats en les consols làsers dels anys 70 i els sintetitzadors visuals dels anys 80.

4.1 Eines de maquinari

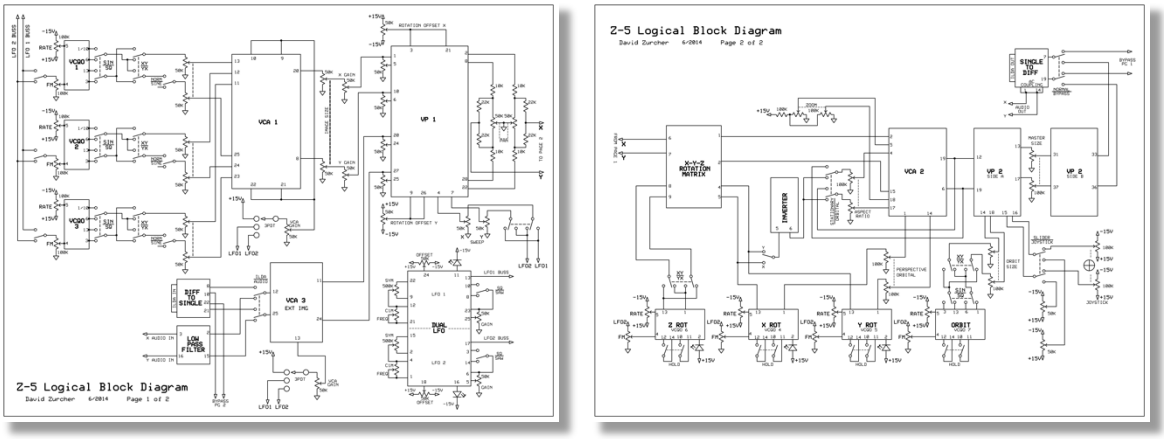
Dins l'oferta actual de maquinari per la creació sono-visual, destaquen els següents dispositius:

Il·lustració 1 - Tauler de control de la consola Z5

4.1.1 Consola Z5 Aldebaraan Systems

L'empresa Aldebaraan Laser Systems ofereix nombrosos productes destinats a oferir mitjans per la creació, processament i connexió de senyals analògics de control per a projectors làsers.

Molts projectors làsers actuals encara fan servir el protocol de comunicacions ILDA (8), dissenyat específicament per enviar senyals des d'un dispositiu capaç de crear voltatges de control. La consola Z5, recentment renovada amb la seva versió Z5+ i el modulador de color Z5c, ofereix les eines habituals en la generació de figures abstractes tan habituals en els primers anys dels espectacles audiovisuals amb llum làser. Com es pot veure en la figura adjunta, el diagrama de blocs de la consola Z5 ha inspirat el disseny del prototip de Quadramat, encara que el nostre disseny incorpora funcionalitats no presents en la primera, i viceversa.



Il·lustració 2 - Diagrama de blocs de la consola Z5

Il·lustració 3 - Cyclops de LXZ Industries

4.1.2 Sintetitzadors modulars en format Euro-rack: LXZ Industries

Els sintetitzadors modulars (9) són aquells en els que els seus components, anomenats mòduls, tenen una entitat separada i es connecten entre ells de forma arbitraria; és a dir, no predeterminada, habitualment formant part d'un sistema que els allotja i proveeix de voltatge d'alimentació, bussos de comunicació, xassís, etc.

Aquests tipus de sintetitzadors es poden trobar en diferents formats, des de els més grans amb unes mides de 5U i connectors de 1/4" com els Moog, MOTM, Hordijk, etc. fins als de mides més reduïdes com el format Euro-rack de mides 3U i connectors de 3.5 mm.

El fabricant LXZ Industries disposa en el seu catàleg de nombrosos mòduls dissenyats específicament per la síntesi d'imatges, i el fet que siguin en format Euro-rack els fa compatibles amb molts mòduls d'altres fabricants, permetent la creació simultània de sons i imatges. Més concretament, fa uns anys vaig contactar amb el soci-gerent d'aquesta companyia i vaig col·laborar en la fase de proves del seu mòdul d'interfície Euro-rack a ILDA, amb el que vaig realitzar algunes proves d'interconnexió de sintetitzadors analògics amb projectors làser. La versió definitiva d'aquest disseny es comercialitza ara amb el nom de Cyclops (10), i permet estalviar mòduls addicionals per crear els voltatges d'offset corresponents als colors RGB, protecció dels "galvos" del projector mitjançant la limitació de l'ample de banda enviat a la interfície ILDA, interruptor de seguretat, sortides diferencials amb les que poder fer servir cables de major longitud, etc.

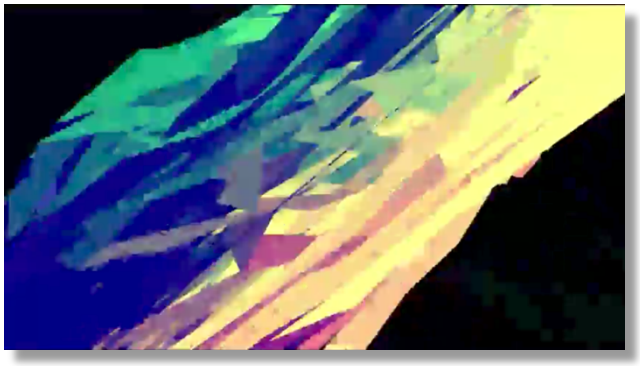
Com es pot veure en la imatge adjunta, el mòdul disposa de cinc entrades de voltatge de control amb connector de 3.5mm. corresponents als senyals X, Y i RGB i una sortida ILDA.

4.2 Eines de programari

Tot i la conveniència i immediatesa de les solucions de maquinari exposades anteriorment, els entorns de disseny sono-visual basats en programari ofereixen una flexibilitat infinitament major ja que la quantitat de tasques que poden executar no depèn d'un conjunt de paràmetres i mòduls preestablerts, sinó que permeten implementar gairebé qualsevol algorisme de generació i processament mitjançant la creació del codi corresponent.

Alguns d'aquests entorns ofereixen la possibilitat de programar el codi fent servir un entorn visual. Així, els fragments de codi (coneguts en alguns entorns amb el nom de *uGens*) es connecten entre ells de manera semblant a com ho fan els mòduls d'un sintetitzador modular. Només que en aquest cas es tracta de cables virtuals i el que representen és un flux de dades de control i àudio i no pas un voltatge real.

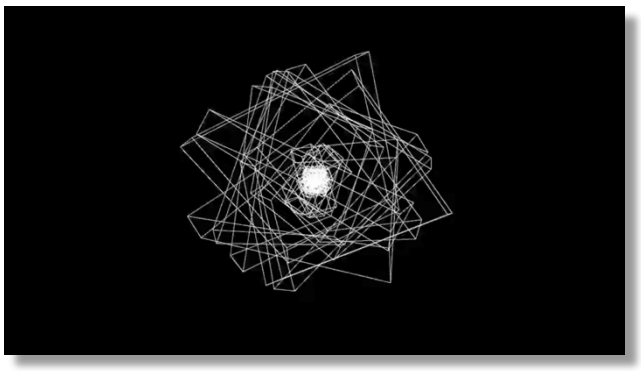
Entre els entorns de programació visual més representatius es troben els següents:



Il·lustració 4 - Visuals en MAX per Infiniteclockworkmachines
https://www.youtube.com/watch?v=k4yYI29HsEw

4.2.1 MAX

Max (11) és l'entorn de programació visual de Cycling74 que permet connectar objectes amb cordons virtuals per crear sons i imatges interactius, i efectes personalitzats. És un dels entorns més coneguts dins l'àmbit acadèmic ja que no només incorpora possibilitats de gestió de dades MIDI, sinó també d'àudio en temps real amb la llibreria MSP, de vídeo amb JITTER, de compilació en temps reals amb GEN, així com llibreries de mòduls de més alt nivell com BEAP i VIZZIE.



Il·lustració 5 – Visuals en Pure Data per Max We
https://www.youtube.com/watch?v=zs1lsw1q_gU&t=71s

4.2.2 Pure Data (EnzienAudio, libpd)

Pure Data (12), també conegut com PD, és una eina de programació visual molt semblant a Max. Les versions inicials d'ambdós programes varen ser desenvolupades per Miller Puckette, encara que el primer és un projecte de codi obert i gaudeix de més acceptació en determinats cercles. Aquesta filosofia de desenvolupament ha fet possible l'aparició de solucions d'interoperabilitat de tercers com libpd o el compilador Heavy de la firma EnzienAudio (13) que permet compilar patches de Pure Data a plataformes tan diverses com Windows, OSX, Linux, PS4, Xbox One, iOS, Android, Bela, Raspberry Pi o, fins i tot, navegadors web mitjançant Javascript. Precisament aquesta darrera opció va ser una de les considerades inicialment per desenvolupar el projecte, però el fet que faci servir ScriptProcessor, un component de la API de Web Audio que ja s'ha declarat obsolet, em va fer decidir per altres alternatives.

4.2.3 Quartz Composer

Quartz Composer (14) és un entorn de programació visual propietat d'Apple i que només funciona en els ordenadors d'aquesta marca. L'ha fet servir la companyia per desenvolupar codi audiovisual interactiu del sistema operatiu OSX, com per exemple protectors de pantalla i visualitzadors d'iTunes. El seu estat de desenvolupament es troba actualment en situació dubtosa, motiu pel qual molts desenvolupadors per aquest entorn l'han abandonat en favor d'altres solucions com Vuo o les esmentades anteriorment.

4.2.4 vvvv

vvvv (15) és un entorn de programació en directe híbrid visual/textual que permet desenvolupament i prototipats senzills. S'ha dissenyat per facilitar el maneig d'entorns de mitjans complexes mitjançant interfícies físiques, gràfics en temps real, i audio i vídeo interactius. Disposa també d'una extensa llibreria de nodes creats per la comunitat d'usuaris. Es tracta d'un programa gratuït per ús no comercial.

Com alternativa a aquests entorns de programació visual, que podrien resultar més atractius pels músics ja familiaritzats amb la síntesi modular, es troben els entorns de programació textual, més habituals en el món de la programació de processat digital del senyal (DSP) i, en general, en qualsevol aplicació de programació convencional. Alguns d'aquests entorns han estat dissenyats específicament per la creació en temps real de sons, i ocasionalment imatges mentre que altres són entorns de programació generalistes que fan servir de llibreries especialitzades per la creació de sons i imatges

Il·lustració 6 - Codi en viu amb Supercollider
https://www.youtube.com/watch?v=_yryiuALe_c

4.2.5 Supercollider

Supercollider (16) és una plataforma per la composició algorítmica i la síntesi d'àudio que fan servir músics, artistes i investigadors sonors. És un programari lliure i de codi obert que es troba disponible per Windows, OSX i Linux. Els seus tres components bàsics són: `scsynth`, el servidor d'àudio en temps real amb més de 400 UGens; `sclang`, un llenguatge de programació interpretat que controla `scsynth` mitjançant Open Sound Control; i `scide`, l'editor per `sclang`.

4.2.6 OpenFrameworks (amb la llibreria Maximilian)

OpenFrameworks (17) és un conjunt d'eines de codi obert per C++ dissenyat per assistir el procés creatiu mitjançant un entorn intuïtiu que afavoreixi l'experimentació. La filosofia de disseny d'aquest conjunt d'eines es pot definir com col·laborativa, usable i simple, consistent i intuïtiva, multi-plataforma, potent i extensible. Per gestionar la programació d'àudio es poden fer servir llibreries com rtAudio, PortAudio, OpenAL and Kiss FFT or FMOD , encara que recentment el Departament d'Informàtica de la *University of London* ha presentat la

llibreria de síntesi d'àudio Maximilian (18), lliure, de codi obert i sota llicencia MIT, amb una sintaxi i estructura de programació semblant al popular entorn de programació Processing, basat en Java.

4.2.7 Processing (amb la llibreria Minim)

Durant els estudis del grau Multimèdia a la UOC, hem tingut vàries oportunitats de fer servir el llenguatge Processing, i a l'assignatura "Tractament i Publicació d'àudio" vàrem fer servir aquest entorn, conjuntament amb la llibreria de síntesi d'àudio Minim, per programar aplicacions de síntesi i anàlisi de so. Processing (19) és un entorn de programació, amb el seu propi IDE (*Integrated Development Environment*) molt adequat per aprendre a programar dins el context de les arts visuals. La llibreria Minim és més completa per aplicacions d'àudio que la que incorpora per defecte el llenguatge Processing i és una molt bona opció per desenvolupar codi audiovisual multi-plataforma.

4.2.8 HTML + CSS + Javascript (amb llibreries p5js + gibber.js, Tone.js, etc.). API Web Audio.

Ens els seus inicis, les possibilitats de generació, i fins i tot reproducció, d'àudio dels navegadors eren molt limitades i, en el cas del navegador Internet Explorer a finals dels anys 90 només permetien la reproducció de l'estàndard conegut com "MIDI files" o arxius MIDI. Aquesta limitació va fer necessari l'ús de plugins com Flash o Quicktime per reproduir continguts sonors però, finalment es va imposar l'estàndard HTML5 i el tag `<audio>` per permetre la reproducció de continguts sonors.

Així i tot, les possibilitats ofertes pel tag `<audio>` són limitades. Sobre tot, pel que fa a la síntesi de sons i al seu processat. L'API Web Audio (20) ha vingut a resoldre algunes d'aquestes limitacions i ofereix noves possibilitats als desenvolupadors, entre les que destaquem la possibilitat d'enrutaments modulars entre els nodes, un model de temporització precís que permet planificar canvis de paràmetres en moments exactes, noves eines de visualització i anàlisi, etc.

https://webaudiodemos.appspot.com/midi-synth/index.html

Il·lustració 7 - Exemple d'app desenvolupada fent servir Web Audio

A les possibilitats ja conegudes del llenguatge de marcatge HTML i d'estilització de CSS, hem d'afegir les que Javascript ofereix per desenvolupar contingut interactiu dins els navegadors. Aquest entorn ha arribat a permetre el desenvolupament d'aplicacions realment tan complexes i funcionals com les aplicacions d'escriptori. Per facilitar la tasca dels desenvolupadors, també han sorgit nombroses llibreries que permeten abstraure part dels més obscurs i complexos detalls inherents a la

programació d'aplicacions més ambicioses. Entre les llibreries que faciliten la tasca d'aprendre a programar amb la mateixa filosofia que l'esmentat entorn Processing trobem P5.js (21), i conjuntament amb gibber.js (22) ofereixen la possibilitat de realitzar senzilles seqüències i processat de senyals sonors. Finalment, entre l'oferta de llibreries existents s'ha escollit la llibreria Tone.js per considerar que és la que més s'acosta a les necessitats del projecte. Els detalls de les seves possibilitats es detallen més endavant en el capítol 11 sobre les APIs utilitzades en el projecte.

5. Continguts

La aplicació consta de tres grans blocs diferenciats. D'una banda, el sintetitzador sono-visual amb la seva corresponent GUI, encarregat de crear els sons i les imatges que són l'objectiu principal de l'aplicació. També forma part d'aquest primer element la matriu de modulació, i la seva missió és repartir els senyals entre els diferents mòduls i cap a l'exterior. D'altra banda, el segon bloc correspon al visualitzador, que és la part encarregada de mostrar la correlació de senyals dret i esquerre, de manera semblant a com un oscil·loscopi representa els senyals en el mode conegut com x/y. Finalment, el tercer bloc el forma la part corresponent als *presets* d'usuari, que permet emmagatzemar i recuperar la posició de cadascun dels paràmetres del sintetitzador.

5.1 Sintetitzador sono-visual

5.1.1 Oscil·ladors

Els mòduls bàsics del sintetitzador són els oscil·ladors. Cadascun d'ells presenta un senyal desfasat noranta graus a les seves dues sortides. D'aquesta manera, s'assegura que amb només un oscil·lador ja es poden obtenir patrons visuals interessants i que la interacció entre ells produeixi imatges encara més complexes.

Els paràmetres dels oscil·lador són els següents:

- Oct, Semi i Cents: fan referència a la Octava, els Semitons i les Centèsimes de semitò, respectivament. El primer es un ajustament gruixat de l'altura tonal, les vuit primeres octaves recauen en el rang subsònic i no generen cap so, però són útils per modular els paràmetres de la resta d'oscil·ladors.
- Waveform: permet escollir la forma d'ona a generar per l'oscil·lador. Les opcions disponibles són sinusoïdal, triangular amb ample de banda limitat als harmònics 3, 5 i 7, triangular completa, quadrada amb ample de banda limitat als harmònics 3, 5 i 7, quadrada completa, dent de serra amb ample de banda limitat als harmònics 2, 3 i 4, dent de serra complet.
- x/y/xy: permet silenciar components del senyal de quadratura seleccionant la possibilitat d'escoltar només el canal esquerre, dret i els dos alhora, respectivament.
- Order: selecciona l'ordre dels polinomis Chebyshev (23) que s'assignaran a la funció de transferència del *waveshaper*. És a dir, si expressem una funció de transferència en forma polinomial, com $y = f(x) = d_0 + d_1x + d_2x^2 + d_3x^3 + \ldots + d_Nx^N$, l'exponent més alt, és a dir N és el que es coneix com l'ordre del polinomi.
- Odd/Even: permet triar si l'ordre dels polinomis Chebyshev és parell o imparell, ja que els dos tipus presenten trets que els diferencien notablement en quant als resultats sonors i visuals que proporcionen quan són aplicats.

5.1.2 Matriu de modulació

La matriu de modulació permet enrutar el senyal d'un oscil·lador cap a les entrades de modulació d'amplitud i freqüència de tots els oscil·ladors disponibles, inclòs ell mateix. Aquesta flexibilitat permet enrutaments amb camins de realimentació que poden resultar enormement complexos i, en ocasions, d'aparença caòtica. No és difícil aconseguir, per exemple, un so semblant al renou blanc aplicant una realimentació de la modulació de freqüència d'un oscil·lador sobre ell mateix. D'altra banda, quan un oscil·lador presenta una freqüència dins el rang subsònic pot crear figures rítmiques i de canvis de l'altura tonal percebuda en intervals pseudo-musicals; mentre que les modulacions en rangs de freqüència audibles generen variacions tímbriques. Així, amb aquesta matriu de modulació es poden controlar variacions sobre els tres paràmetres fonamentals del so: amplitud, freqüència i timbre.

5.2 Visualitzador

Habitualment es fan servir dispositius com oscil·loscopis per observar com un senyal varia en el temps. El mode més habitual d'aquest tipus de dispositius és el de visualització de la forma d'ona del senyal, en el que un feix de llum va desplaçant-se horitzontalment segons una base de temps predefinida i verticalment segons el valor instantani del senyal en cada moment.

Però també es pot visualitzar el senyal en un mode alternatiu conegut com xy en el que el senyal present a un canal del dispositiu s'encarrega del desplaçament horitzontal del senyal i el senyal present al segon canal del desplaçament vertical. Així es pot fer una visualització de la correlació de fase, freqüència i amplitud entre els dos senyals. El visualitzador implementat és d'aquest segon tipus i mostra simultàniament els senyals dels canals dret i esquerre del sintetitzador en forma de moviment horitzontal i vertical, respectivament. També s'ha simulat la persistència típica dels oscil·loscopis analògics, ja que ajuda a seguir el desplaçament del feix de llum a baixes freqüències.

5.3 Sistema de presets

La finalitat del sistema de *presets* és facilitar als usuaris la recuperació de programacions complexes per poder repetir-les sense haver de recordar el valor de cadascun dels nombrosos paràmetres del sintetitzador. Els primers sintetitzadors analògics no disposaven d'aquesta funcionalitat i, com alternativa, els usuaris feien servir fulls de programació, o plantilles de paper amb la interfície de l'instrument sobre les que anotaven els valors dels paràmetres.

Afortunadament, per emmagatzemar el valor de tots els paràmetres només cal implementar una senzilla base de dades en la que cada camp es correspon amb un paràmetre del sintetitzador. Els usuaris es poden donar d'alta amb un e-mail, que s'ha de confirmar posteriorment, i ja donats d'alta en el sistema, poden guardar les programacions donant-les un nom per identificar-les.

6. Metodologia

La metodologia que s'aplicarà en el procés de desenvolupament del programari corresponent al projecte ens haurà de garantir que el cicle de desenvolupament i la seva divisió en fases diferenciades ajuden a millorar el disseny, la gestió del producte i la gestió de la gestió de tot el procés. (24)

La metodologia que s'ha decidit seguir en el desenvolupament del projecte que ens ocupa és la coneguda com **model de prototips** (25) i ens proposa la creació de prototips d'aplicacions des de zero o des de versions incompletes de manera semblant a com es fa en els camps de l'enginyeria mecànica o la manufactura.

Els principals avantatges e inconvenients d'aquesta metodologia es descriuen en la següent taula.

Avantatges
Reducció del temps i dels costos
Requereix la participació dels usuaris mitjançant la seva interacció obligada, el que pot donar una millor solució a les necessitats plantejades
En el cas del projecte actual, permetrà descobrir la funcionalitat disponible en el propi procés de prototipat
Inconvenients
Anàlisi insuficient respecte d'una visió global del projecte, ja que la transformació dels prototips incomplets o ineficients pot resultar difícil de mantenir
Confusió entre els usuaris i el desenvolupador, que en aquest cas concret serà evitada ja que el desenvolupador en serà el principal usuari i només en les fases finals del prototip es faran servir proves externes
Afecció del desenvolupador a una versió incompleta del prototip, deguda al temps que hi ha invertit. Per evitar aquest inconvenient, s'ha de tenir clar des del primer moment que es seguirà un procés de prototipat evolutiu
Risc d'incórrer en excessiu temps i costos si no es fixen objectius realistes i assolibles.

Taula 1 - Avantatges e inconvenients del model de prototips

El motiu principal que m'ha portat a escollir aquesta metodologia de desenvolupament és la potencial reducció de temps i costos, ja que la programació de sistemes complexes de disseny sonor moltes vegades fa servir aquest mateix tipus de metodologia i ja n'estic familiaritzat. A més, permetrà anar descobrint la funcionalitat de les llibreries a fer servir.

Els prototips aniran implementant progressivament tota la funcionalitat prevista, i cadascun d'ells serà operatiu des del primer moment.

S'espera anar desenvolupant els següents prototips de forma iterativa:

1. *Front-end* bàsic amb prova de concepte sobre oscil·ladors de quadratura, amb desfasament constant de noranta graus, i visualització de la correlació de fase de les dues sortides formant una corba de Lissajous

2. Aplicació del patró de disseny modular per millorar l'estructura i la llegibilitat del codi. Ampliació del nombre d'oscil·ladors i de tots els seus paràmetres corresponents. Finalització de la matriu de modulació.

3. Creació dels sistema de *presets* i connexió amb el *back-end* (servidor). Darrers retocs de la interfície gràfica.

4. Versió definitiva completament funcional

7. Arquitectura de l'aplicació

7.1 Aplicació

L'aplicació crea els sons i les imatges mitjançant la manipulació dels elements de la GUI. Podem dividir la seva funcionalitat en quatre blocs diferenciats: codi HTML amb la disposició dels elements que configuren la GUI i el visualitzador, codi Javascript corresponent al motor de síntesi, codi Javascript corresponent a la interfície d'usuari, i codi corresponent al visualitzador de figures Lissajous.

7.1.1 Motor de síntesi

Els elements bàsics de l'aplicació de síntesi són els nodes, també coneguts en altres entorns com UGens, i més específicament en l'API de Web Audio amb el nom d'AudioNodes. Tone.AudioNode és la classe base per totes les classes de la llibreria Tone.js que processen àudio.

El motor de síntesi permet la interconnexió d'aquests nodes entre sí, enrutant els senyals que generen i/o processen.

Alguns exemples dels nodes que es fan servir en el desenvolupament del motor de síntesi són els següents:

- `Tone.Gain`: node bàsic per enrutar senyals i ajustar el seu guany
- `Tone.Oscillator`: generador bàsic de so
- `Tone.Merge`: mesclador per dos senyals en un senyal bi-canal.
- `Tone.Waveform`: obté les dades de la forma d'ona per la seva visualització
- `Tone.Chebyshev`: waveshaper que fa servir polinomis de Chebyshev

7.1.2 Interfície d'usuari

La llibreria NexusUI proporciona interfícies HTML, i fent servir les funcions de Javascript que permeten l'execució de codi arbitrari quan l'usuari interactua amb els elements de la GUI, creats a partir de gràfics en format SVG, es poden canviar els valors dels paràmetres oportuns. A cada paràmetre li correspon el seu element dins la interfície, o fins i tot en el cas de la freqüència de l'oscil·lador, aquesta s'obté a partir de la combinació dels valors de tres elements de tipus "botó".

7.1.3 Visualitzador de figures Lissajous

Es fa servir un element `canvas` sobre el que es dibuixen línies en correspondència als valors dels senyals dels canals dret i esquerre del sintetitzador que es llegeixen de dos nodes `Tone.Waveform` prèviament definits.

7.2 Servidor

La seva funció és servir l'aplicació (HTML, CSS, JS) mitjançant node.js i el *framework* Express.js. Entre les funcions de Node.js hi ha les de generar contingut dinàmic i també les de connexió amb la base de dades MongoDB, llegint-ne, esborrant i modificant-hi dades corresponents als presets (reglatges dels valors dels paràmetres del sintetitzador)

7.3 Base de dades

En el projecte s'ha fet servir MongoDB (26), una base de dades no-relacional que suporta el llenguatge de consulta JSON i, en comptes de està basat en taules com SQL es basa en col·leccions i parells clau-valor.

Aquest tipus de base de dades s'adapta molt bé a les necessitats del projecte, que no presenta una estructura amb relacions dels seus elements *many-to-many* i que permet des de Javascript la manipulació i assignació dels valors emmagatzemats de forma molt senzilla.

Els documents de Mongo també comparteixen la seva estructura amb els objectes en llenguatges com Javascript, ja que són una mena de JSON, el que facilita relacionar les dades que es fan servir a l'aplicació amb les que s'emmagatzemen a la base de dades.

Fent servir MongoDB, amb els valors i posició de cada paràmetre es crea un preset, que serà públic i es podrà compartir. L'estructura de la base de dades serà senzilla, ja que cada preset serà un JSON amb tota la informació necessària.

Existeixen tres col·leccions (en terminologia de MongoDB) i cadascuna s'encarrega de gestionar les dades corresponents als presets, als usuaris i a les sessions d'usuari, respectivament.

En l'annex 2 s'inclouen exemples i extractes de les bases de dades de l'aplicació.

El diagrama de l'arquitectura de l'aplicació es troba a la *Il·lustració 18 - Diagrama de l'arquitectura de l'aplicació*

8. Plataforma de desenvolupament

8.1 Programari

8.1.1 Brackets

La principal eina de desenvolupament emprada en el projecte és l'editor de codi Brackets, una iniciativa impulsada per Adobe però de codi lliure. Entre les seves característiques destaquen la possibilitat d'editar codi en viu, ja que qualsevol canvi al codi HTML o CSS es pot veure reflectit immediatament al navegador. També permet visualitzar de forma senzilla sobre quins elements presents al navegador s'aplica un CSS només fent clic sobre els selectors oportuns.

A més, permet afegir-hi funcionalitat addicional mitjançant l'ús d'extensions. Entre d'elles, pel projecte actual s'han fet servir les extensions Brackets Beautify, per indentar el codi automàticament; Brackets Git, per integrar Git dins l'editor; i Brackets Synapse, que permet la sincronització d'arxius entre l'ordinador local i un servidor remot via FTP o SFTP.

8.1.2 GitHub

El servei de compartició i publicació de codi GitHub (26) fa servir i proporciona una interfície web gràfica per Git, el projecte de codi obert iniciat per Linus Torvalds. Git és un sistema de control de versions, que gestiona i emmagatzema revisions de projectes. Es podria dir, en paraules senzilles, que és una manera de gestionar els esborranys dels documents.

A causa de la naturalesa individual del projecte que ens ocupa, no s'han fet servir característiques essencials de Git com *"forking"*, *"pull request"* o *"merge"*; però així i tot, només el fet de fer servir GitHub ha ajudat a estructurar i millorar el procés de planificació del desenvolupament, oferint alhora una important eina per revisar el codi i confirmar que el ritme de treball s'ajusta a les previsions inicials.

8.1.3 Draw.io

Per l'elaboració dels diagrames de blocs del sintetitzador s'ha fet servir l'aplicació web draw.io. Aquest programari de diagrames permet realitzar online i de forma gratuïta diagrames de flux, diagrames de processos, UML, ER i diagrames de xarxa.

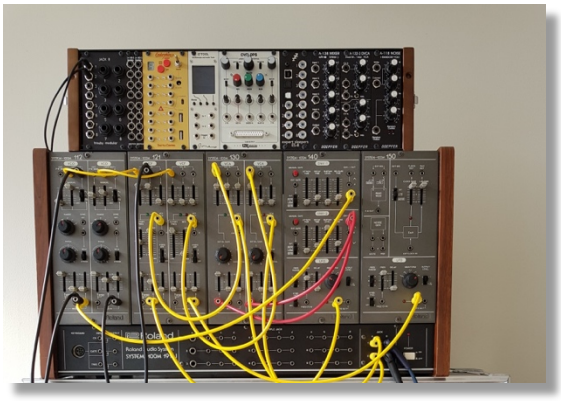
8.2 Maquinari

8.2.1 Ordinador amb OSX

Per les tasques de desenvolupament del codi s'ha fet servir un ordinador Mac Pro dotat del sistema operatiu macOS Sierra (10.12)

8.2.2 Interfície d'àudio + ILDA (connexions modular)

Per generar els sons s'ha fet servir la tarja de so integrada de l'ordinador, ja que les seves característiques resulten suficients per dur a terme el projecte actual. La sortida de línia de l'ordinador es connecta a una taula de mescles amb dues sortides estereofòniques: una sortida mestra que es connecta als altaveus (Genelec 8040) o i una altra de monitoratge que es connecta al sintetitzador modular.



Il·lustració 8 - Sintetitzador modular del projecte

La funció del sintetitzador modular és, en aquesta ocasió, només la de proporcionar una interfície adequada amb el projector làser i una confirmació visual dels senyals rebuts. Així, el senyal creat per l'aplicació surt de l'ordinador, passa per la taula de mescles, entra dins el visualitzador de maquinari *O'Tool* del sintetitzador modular, des d'on passa a la interfície Cyclops, que no només permet canviar el tipus de connector de mini-jack de 3.5mm al D-SUB-25 que fa servir la interfície ILDA, sinó que també genera els voltatges necessaris per controlar la intensitat dels tres canals de color RGB i altres transformacions com inversió i escalament dels senyals, balancejat, filtratge, etc.

8.2.3 Projector làser

Il·lustració 9 - Projector làser Swisslas PM-1400 RGB

El projector que s'ha fet servir en el projecte és un Swisslas PM-1400 RGB amb una potència de 1-25W-1.5W, depenent del color. Pertany a la família dels projectors làsers per espectacles audio-visuals i avui en dia es fa servir normalment en combinació amb programari especialitzat com QuickShow o Beyond. Aquest tipus de projector fa servir galvanòmetres, també coneguts com escàners o galvos, que són uns dispositius electromagnètics que desplacen uns miralls on es reflecteix el feix de llum generat per uns díodes, permetent així posicionar el feix de llum i generar figures quan aquest es desplaça a molt velocitat. El projector esmentat disposa d'interfície ILDA, que presenta un connector D-SUB-25 pel qual rep els voltatges que controlen la posició del feix de llum a generar i la intensitat dels colors que el formen.

9. Planificació

9.1 Dates clau

La planificació del projecte es troba condicionada a les dates d'entrega fixades en el pla docent de l'assignatura. Així trobem les següents dates clau preestablertes:

Títol	Inici	Lliurament	Objectius
PAC 1	20/09/2017	03/10/2017	Definició formal del projecte i la pauta de treball a seguir
PAC 2	04/10/2017	01/11/2017	Començament investigació i desenvolupament. Revisió de requisits, mètodes de treball i eines. Consolidació fonaments del projecte: teòrics, model base de dades, *wireframes* d'interfícies, etc.
PAC 3	02/11/2017	03/12/2017	Gruix de treball del projecte. Llançament de la versió beta
Lliurament final	06/12/2017	14/01/2018	Finalització del projecte, la seva documentació i publicació del treball

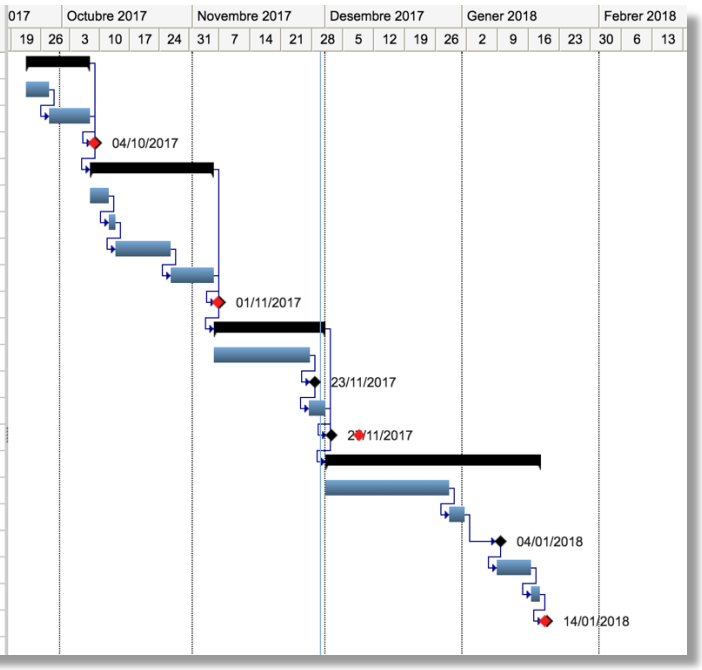
Taula 2 - Dates clau preestablertes

9.2 Relació de fites del projecte

- 03/10/2017 - PAC 1: Definició formal del projecte
- 01/11/2017 - PAC 2: Entrega del prototip funcional amb oscil·ladors i matriu de modulació
- 26/11/2017: Llançament de la versió beta del motor d'àudio
- 03/12/2017 – PAC 3: Llançament de la versió beta amb sistema de preajustaments i connexió amb el servidor
- 04/01/18: Llançament de la versió definitiva
- 14/01/18 – Lliurament final: Llançament de la versió definitiva, documentació i publicació

9.3 Diagrama de Gantt

	Nom	Inici	Fi
1	**FASE 1 - Definició formal del projecte**	**20/09/2017**	**04/10/2017**
2	Proposta del tema al consultor	20/09/2017	25/09/2017
3	Definició formal projecte i pauta de treball	25/09/2017	04/10/2017
4	Entrega del document de la PAC 1	04/10/2017	04/10/2017
5	**FASE 2 - Prototip funcional**	**04/10/2017**	**01/11/2017**
6	Investigació API i llibreries ToneJS i NexusUI	04/10/2017	08/10/2017
7	Creació Wireframe interfície gràfica	08/10/2017	10/10/2017
8	Desenvolupament Prototip 1: front-end bàsic amb 1 os	10/10/2017	23/10/2017
9	Preparació del document de la PAC 2	23/10/2017	01/11/2017
10	Entrega del document de la PAC 2	01/11/2017	01/11/2017
11	**FASE 3 - Gruix del treball i versió beta**	**01/11/2017**	**27/11/2017**
12	Desenvolupament Prototip 2: aplicació patró disseny m	01/11/2017	23/11/2017
13	Llançament de la versió beta	23/11/2017	23/11/2017
14	Preparació del document de la PAC 3	23/11/2017	27/11/2017
15	Entrega del document de la PAC 3	27/11/2017	27/11/2017
16	**FASE 4 - Finalització del projecte**	**27/11/2017**	**14/01/2018**
17	Desenvolupament Prototip 3: sistema de presets i conn	27/11/2017	25/12/2017
18	Desenvolupament versió definitiva: correcció de bugs i	25/12/2017	28/12/2017
19	Llançament de la versió definitiva	04/01/2018	04/01/2018
20	Finalització de la documentació	04/01/2018	12/01/2018
21	Publicació	12/01/2018	14/01/2018
22	Lliurament final del document de la PF	14/01/2018	14/01/2018



Il·lustració 10 - Diagrama de Gantt del projecte

Document font en línia:
https://drive.google.com/file/d/0B3kDZHKgxM7DRXhjN3prbXhtU00/view?usp=sharing

9.4 Relació de tasques

FASE 1 - Definició formal del projecte - 20/09/2017 fins 04/10/2017

- Proposta del tema al consultor - 20/09/2017 fins 25/09/2017
- Definició formal projecte i pauta de treball - 25/09/2017 fins 04/10/2017
- *Entrega del document de la PAC 1 - 04/10/2017*

FASE 2 - Prototip funcional - 04/10/2017 fins 01/11/2017

- Investigació API i llibreries ToneJS i NexusUI - 04/10/2017 fins 08/10/2017
- Creació Wireframe interfície gràfica - 08/10/2017 fins 10/10/2017
- Desenvolupament Prototip 1: front-end basic - 10/10/2017 fins 23/10/2017
- Preparació del document de la PAC 2 - 26/10/2017 fins 01/11/2017
- *Entrega del document de la PAC 2 - 01/11/2017*

FASE 3 - Gruix del treball i versió beta - 01/11/2017 fins 27/11/2017

- Desenvolupament Prototip 2: aplicació patró disseny modular, 4 oscil·ladors i matriu de modulació completa - 01/11/2017 fins 23/11/2017
- Llançament de la versió beta - 23/11/2017
- Preparació del document de la PAC 3 - 23/11/2017 fins 27/11/2017
- *Entrega del document de la PAC 3 - 27/11/2017*

FASE 4 - Finalització del projecte - 27/11/2017 fins 14/01/2018

- Desenvolupament Prototip 3: sistema de presets i connexió amb back-end - 27/11/2017 fins 31/12/2017
- Desenvolupament versió definitiva: correcció de *bugs* i millores usabilitat - 31/12/2017 fins 04/01/2018
- Llançament de la versió definitiva: 04/01/2018
- Finalització de la documentació - 04/01/2018 fins 12/01/2018
- Publicació - 12/01/2018 fins 14/01/2018
- *Lliurament final del document de la PF - 14/01/2018*

10. Procés de treball/desenvolupament

10.1 Tria de les tecnologies i llibreries del projecte

10.1.1 Front-end: HTML + CSS + JS

- Bootstrap: http://getbootstrap.com/
- ToneJS: https://tonejs.github.io/
- NexusUI: https://nexus-js.github.io/ui/

Inicialment es va considerar fer servir la llibreria P5.js i la seves llibreries integrades P5.sound i P5.gui, però després d'algunes proves es va descartar per no disposar de tota la funcionalitat necessària per implementar els complexos enrutament de senyal que requereix l'aplicació.

La següent taula mostra una comparativa dels punts forts i febles de les dues opcions considerades:

	Punts forts	**Punts febles**
p5js + p5.sound + p5.gui	integració directa de les llibreries de so i interfície amb la llibreria principal comunitat d'usuaris activa similitud amb Processing, que hem estudiant durant el grau IDE específic amb servidor web local integrat	p5.sound sembla no permetre enrutaments modulars complexos (bussos, *merge*, *split*, etc.) p5.gui amb funcionalitat més reduïda
tone.js + nexus.ui	tone.js es troba en un estat de desenvolupament molt més madur i incorpora més funcionalitat de Web Audio tone.js permet enrutaments modulars amb facilitat (bussos, *merge*, *split*, etc.) nexus.ui es troba molt més ben adaptat als dispositius tàctils nexus.ui ha estat dissenyat específicament per interfícies d'àudio	major complexitat de les llibreries

Taula 3 - Comparatiu P5js amb Tone.js + NexusUI

10.1.2 Back-end: MEAN

3. Servidor web Node.js https://nodejs.org/en/ amb l'entorn de treball Express.js http://expressjs.com/es/ . Encara que PHP ens ofereix de sobra tota la funcionalitat requerida i és un entorn més familiar en els estudis del grau, he decidit provar Node.js amb el mòdul express.js ja que la seva popularitat augmenta per moments i no s'ha de canviar de llenguatge respecte del front-end, ja que es continua fent servir Javascript.

4. Sistema Gestor de Base de Dades: https://www.mongodb.com/es. MongoDB és una base de dades documental que emmagatzema les dades en estructures definides en notació JSON i que resulta molt adequat per emmagatzemar els presets de Tone.js

5. Plataforma: https://www.heroku.com/

   - Contenidor NodeJS: https://www.heroku.com/nodejs
   - Addon MongoDB: https://elements.heroku.com/addons/mongolab

La plataforma heroku ens proporciona els dos serveis que necessitem, que són Node.js i MongoDB, sense necessitat de configurar un servidor, facilitant la feina ja que ens abstreu de la part de "sistemes" (sistema operatiu, instal·lació, configuració, manteniment...)

La següent taula mostra una comparativa dels punts forts i febles de les dues opcions considerades:

	Punts forts	**Punts febles**
LAMP: Linux, Apache, MySQL, PHP	més conegut i usat major arrelament en la comunitat suportat per gairebé tots els proveïdors	cal fer servir llenguatges diferents pel front-end i pel back-end menys ràpid
MEAN: MongoDB, Express.js, Angular.js, Node.js	tot es programa en un únic llenguatge major rapidesa i escalabilitat el sistema de base de dades s'adequa millor a les necessitats del projecte	menys conegut cal fer servir un proveïdor més "especialitzat" menor suport per part dels proveïdors

Taula 4 - Comparatiu back-end LAMP amb MEAN

10.2 Prototip #1: *Front-end* bàsic amb un oscil·lador

La creació del primer prototip implica una tasca d'investigació inicial per conèixer suficientment les característiques de les tecnologies i llibreries relatives al *front-end*, és a dir Tone.js i NexusUI.js, així com experimentar amb els mètodes de generació sonora que permetran aconseguir els resultats sonors i visuals esperats.

10.2.1 Tipus de valors i senyals de Tone.js

En primer lloc ha calgut investigar sobre els tipus de valors i senyals amb que fa feina Tone.js i esbrinar els seus rangs per poder preveure l'estructura de guanys (*gain staging*) a fer servir i les eventuals necessitats de conversió de valors. D'aquest procés es varen obtenir les següents dades rellevants al funcionament de Web Audio i, més concretament, alguns dels tipus ("types") definits (27) per la llibreria Tone.js.

- **Gain**: Relació entre l'entrada I la sortida d'una senyal. Un gain de 0 silencia la señal, un guany de 1 no causa cap canvi. NOTA: Provar si un guany de -1 inverteix la fase del senyal per poder implementar modulació d'amplitud bipolar
- **Decibels**: unitat de mesura logarítmica útil per el volum a causa de la manera en que percebem la sonoritat. 0 significa cap canvi en volum, -12 correspon a la meitat de fort
- **AudioRange**: els valors d'AudioRange són entre [-1, 1].
- **NormalRange**: els valors de NormalRange són entre [0, 1].
- **Hertz**: Representació de la freqüència en cicles per segon
- **Degrees**: Angle entre 0 i 360

10.2.2 Connexions entre mòduls

Els nodes de la llibreria Tone.js es connecten pràcticament igual que els de Web Audio, fet que és normal degut a que la llibreria està construïda sobre l'API i, en algunes ocasions, només "empaqueta" les seves funcions. Les connexions entre els nodes es realitzen de la següent manera (28):

- D'un origen a una destinació:

```
origen.connect(destination);
```

- Una cadena de connexions en sèrie:

```
origen.chain(intermediDesti, intermediDesti, finalDesti);
```

- D'un node a varies destinacions en paral·lel:

```
origen.fan(desti1, desti2);
```

10.2.3 Waveshaping per polinomis Chebyshev:

La funcionalitat de Waveshaping per polinomis Chebyshev no es gaire habitual en els sintetitzadors i es considera una tècnica avançada. Després d'observar que la llibreria Tone.js la incorporava i preveient les seves possibles aplicacions per modificar les formes

d'ona bàsiques proporcionades pels oscil·ladors, vaig realitzar una petita tasca de recerca per esbrinar com es podia aplicar. Després d'algunes proves, els resultats varen ser molt satisfactoris tot i que queda pendent d'experimentar amb les possibilitats addicionals que dos controls de guany podrien proporcionar, un a la entrada del *waveshaper* augmentant el nivell del so i un a la sortida reduint-lo en proporció inversa.

Finalment es va decidir aplicar un waveshaper de tipus Chebyshev a cada oscil·lador.

10.2.4 Assignació de freqüències exponencial (en termes musicals)

L'assignació de la freqüència a cada oscil·lador va patir canvis considerables fins arribar a la versió actual.

La primera implementació feia servir només un botó per assignar la freqüència de l'oscil·lador i això presentava certs problemes a causa de les dificultats de controlar amb exactitud la freqüència donat el reduït rang de valors possibles en un gir del botó. A més, aquest botó feina servir una escala lineal, proporcionant un rang d'octaves desigual en el seu recorregut.

Això va portar a la idea de fer servir més d'un botó per assignar la freqüència i passar a valors exponencials, com els que fan servir les notes musicals. Un per seleccionar l'octava, un altre per seleccionar els semitons i un tercer per les centèsimes de semitò. Aquesta configuració aconsegueix un control exacte dels valors que té molt sentit des del punt de vista dels processos a realitzar mitjançant la interacció dels valors de les freqüències dels diferents oscil·ladors.

Es va descartar fer servir la funcionalitat de la llibreria NexusUI per canviar la freqüència arrel ("*root*"). Per defecte la freqüència arrel és Do central, 261.62556530059Hz, i es pot canviar fent servir `Nexus.tune.root` de les següents maneres:

```
Nexus.tune.root = Nexus.note (0,-3);// 3 octaves per davall del seu valor actual
Nexus.tune.root = Nexus.note (-1); // un semitó per davall del seu valor actual
Nexus.tune.root = 120; // un valor en Hz determinat
```

Una altra qüestió era la de millorar la relació de freqüències entre els diferents intervals. Tot i que la afinació temperada, en la que cada semitò té $\sqrt[12]{2}$ vegades la freqüència de l'anterior és òptima per objectius musicals més generals, en el nostre cas calia fer servir una escala justa (29) en el paràmetre semitons per aconseguir unes relacions de freqüència exactes que no generin batiments originats per la lleugera desviació dels valors assignats per la escala temperada respecte del valors fraccionals obtinguts amb l'escala justa.

Finalment, es va fer servir el mètode Nexus.tune.createJIScale() per definir els valors de la escala justa del sintetitzador.

```
Nexus.tune.createJIScale( // Escala justa
8 / 15,
9 / 16, // RE -1
3 / 5,
81 / 128, // MI -1
2 / 3, // FA -1
729 / 1024,
3 / 4, // SOL -1
4 / 5,
```

```
27 / 32, // LA -1
8 / 9,
243 / 256, // SI -1
1 / 1, // DO
256 / 243,
9 / 8, // RE
32 / 27,
5 / 4, // MI
4 / 3, // FA
1024 / 729,
3 / 2, // SOL
128 / 81,
5 / 3, // LA
16 / 9,
15 / 8); //SI
```

10.2.5 Modulació en freqüència lineal (Linear FM). Thru zero.

La modulació de freqüència que apliquen els oscil·ladors a altres oscil·ladors podria ser de tipus lineal o exponencial. En el cas de la modulació de freqüència lineal, si el valor central de la freqüència generada per l'oscil·lador és 1000Hz, una modulació de freqüència lineal de 500 Hz crearia valors entre 500Hz i 1500Hz. En canvi una modulació de freqüència exponencial de "una quinta" crearia valors entre 666,6Hz i 1500Hz.

Ja que la modulació de freqüència lineal és millor per modificar espectre sonor i la exponencial per *vibrato*, es va escollir implementar modulació de freqüència lineal enviant directament els valors generats per l'oscil·lador al bus `oscNFreqChan` que rep cada oscil·lador mitjançant:

```
OscNx.frequency.receive("oscNFreqChan")
```

S'hauria pogut aconseguir una modulació exponencial de la freqüència dels oscil·ladors fent servir `OscNx.detune.receive("oscNFreqChan")`, que afecta a la freqüència de l'oscil·lador segons un valor rebut en centèsimes de semitò i, per tant, exponencial.

Cal destacar el funcionament dels oscil·ladors de la llibreria Tone.js quan reben una modulació que els fa adquirir valors de freqüència negatius, ja que en aquest cas la freqüència resultant és equivalent al valor absolut que pertocaria, però amb una inversió de polaritat.

10.2.6 Modulació en amplitud (AM)

Existeixen dos tipus diferents de modulació en amplitud, la de tipus unipolar i la bipolar (30). Com en el sintetitzador es fa servir un senyal bipolar, la sortida d'un oscil·lador, per modular l'amplitud dels altres, el resultat s'identifica amb una modulació en anell, en la que un valor negatiu del senyal del modulador inverteix la polaritat de la portadora.

Després de realitzar proves, es decideix no escalar el senyal de la moduladora per permetre una vertadera modulació en anell.

10.2.7 Proves d'usabilitat de la Interfície d'usuari

En els primers experiments es va fer servir la versió anterior de NexusUI, però les millores aparegudes en la versió actual varen fer recomanable l'actualització.

Unes primeres proves d'usabilitat varen servir per determinar els ajustaments dels botons de la GUI i situar-los en el mode d'interacció "vertical", en el que l'usuari canvia el valor del botó amb un desplaçament vertical del dit o el punter del ratolí. També es va optar per fer servir el mode "relative", i així evitar els canvis bruscos dels paràmetres.

10.2.8 Taula de la matriu de modulació

En la confecció de la taula del primer prototip es va optar per fer servir una taula en el marcatge HTML ja que presenta no només una manera convenient de disposar els elements sinó també una opció correcta des del punt de vista semàntic.

10.2.9 Experiments descartats

Durant la fase d'investigacions i proves del primer prototip es va considerar i descartar la possibilitat de implementar sincronització dels oscil·ladors. Entre els punts positius figura la possibilitat de crear patrons de modulació interessants, però fer-ho de manera productiva hauria incrementat sensiblement la complexitat del prototip i canvis en la GUI que van desaconsellar implementar-ho de moment.

També es va descartar aplicar un control de fase manual entre els dos components de l'oscil·lador per no oferir els resultats sonors i visuals esperats, així com un control de desafinació dels dos components d'un oscil·lador, pels mateixos motius.

10.3 Prototip #2: Disseny modular, oscil·ladors i matriu de modulació

10.3.1 Patró de disseny modular.

Segons l'autor Addy Osmani, en el seu llibre *Javascript Design Patterns* (31), "Els mòduls són una peça integral de qualsevol arquitectura de l'aplicació sòlida i normalment ajuden a mantenir les unitats de codi per a un projecte tan netament separat i organitzat .../.../... El patró de mòduls es va definir originalment com una forma d'oferir encapsulació privada i pública per a les classes en enginyeria de programari convencional. A Javascript, el patró de mòduls s'utilitza per seguir emulant el concepte de classes de manera que podem incloure tant mètodes públics com privats i variables dins d'un sol objecte, protegint així parts concretes de l'àmbit global. El que això implica és una reducció de la probabilitat que els nostres noms de funcions es trobin en conflicte amb altres funcions definides en scripts addicionals a la pàgina."

Es podria dir que els bons programadors divideixen el seu codi en mòduls de la mateixa manera que els bons autors literaris divideixen les seves obres en capítols i seccions (32) i, d'aquesta manera els bons mòduls es pot dir que són altament auto-continguts i amb funcionalitat diferenciada, la qual cosa permet intercanviar-los, eliminar-los o afegir-los, si és necessari, sense interrompre el sistema a nivell global.

Entre els beneficis de la modularització del codi destaquem els següents:

- Facilitat de manteniment: és més fàcil actualitzar un mòdul si es troba desacoblat dels altres elements del codi
- Creació d'espais de noms: ja que a Javascript les variables que es troben fora de l'àmbit de la funció de nivell superior es consideren globals, la modularització ajuda a evitar la contaminació dels espais de noms, creant un espai privat per les variables
- Reusabilitat: permet reaprofitar el codi en altres projectes.

Un mòdul compleix dos propòsits: Primer, conté contingut, assignant identificadors a valors. En segon lloc, proporciona un espai de noms per a aquests identificadors, per evitar que col·lisionin amb identificadors en altres mòduls. A JavaScript, els mòduls s'implementen a través d'objectes.

Per definir l'espai de nom, un mòdul de nivell superior es col·loca en una variable global. Aquesta variable és l'espai de noms del contingut del mòdul. En el nostre cas, ho implementem de la següent manera:

```
var qm = {};
```

Per niar submòduls, senzillament apliquem el patró de mòduls de nou i definim nous objectes dins l'objecte anterior.

```
qm.engine = {}

qm.ui = {}

qm.ui.e = {}

qm.ui.e.dials = {}
```

Posteriorment, cada mòdul disposa del seu propi contingut, accessible als altres mòduls fent ús del subespai de noms corresponent. Així, per exemple, el mòdul encarregat de la interfície d'usuari pot accedir tant al seu propi contingut per llegir o modificar els valors dels seus elements, com al contingut del motor d'àudio (*audio engine*) per modificar els paràmetres dels nodes de Tone.js.

```
qm.ui.changes = {
    dialOsc1OctSemiCents: function(v, dial) {
    if(dial) dial.value = v
    qm.ui.assignaValorFreq(
        qm.ui.e.dials.dialOsc1Oct,
        qm.ui.e.dials.dialOsc1Semi,
        qm.ui.e.dials.dialOsc1Cents,
        qm.engine.nodes.osc1a,
        qm.engine.nodes.osc1b);
    }
}
```

10.3.2 Implementació de la resta d'oscil·ladors

Una vegada aconseguit un oscil·lador completament funcional, la resta resulta més fàcil d'implementar gràcies als esforços invertits en la modularitat del codi.

Ja que una de les màximes de la bona programació és seguir el model DRY *("Don't Repeat Yourself"*) (33), per aconseguir aquest objectiu es programen algunes funcions que eviten la repetició innecessària de llargs fragments de codi idèntics per cada oscil·lador. A aquestes funcions, entre d'altres, se'ls passa com un paràmetre d'entrada l'objecte corresponent a l'oscil·lador sobre el que es vol actuar.

Es detallen a continuació alguns exemples de l'aplicació d'aquest principi en la nostra aplicació:

```
qm.ui.assignaIndex = function(v, chebyOscA, chebyOscB) {
    chebyOscA.input.gain.linearRampToValueAtTime(v, Tone.now() + 0.05);
    chebyOscB.input.gain.linearRampToValueAtTime(v, Tone.now() + 0.05);
    chebyOscA.output.gain.linearRampToValueAtTime(1 / v, Tone.now() + 0.05);
    chebyOscB.output.gain.linearRampToValueAtTime(1 / v, Tone.now() + 0.05);
};
```

En aquest exemple, per assignar l'índex de modulació del *waveshaper* de cada oscil·lador, el gestor d'events de la GUI crida la funció enviant-li tant el valor llegit a l'element de la GUI com els oscil·ladors sobre els que ha d'aplicar aquest valor. També podem observar en aquest fragment de codi que, per evitar els *clicks* resultants d'un canvi brusc de valors, es fa servir el mètode `linearRampToValueAtTime()` que rep com arguments el valor final al que es vol canviar i la durada de la transició del valor actual al valor final; en el nostre cas, el valor canvia en el transcurs de 50 mili-segons, un valor suficient fins i tot per freqüències molt baixes.

```
qm.ui.assignaXyType = function(v, oscA, oscB) {
    if (v.value == "xy") {
        oscA.volume.linearRampToValueAtTime(-6, Tone.now() + 0.06);
        oscB.volume.linearRampToValueAtTime(-6, Tone.now() + 0.06);
    } else if (v.value == "x") {
        oscA.volume.linearRampToValueAtTime(-Infinity, Tone.now() + 0.06);
        oscB.volume.linearRampToValueAtTime(-6, Tone.now() + 0.06);
    } else if (v.value == "y") {
```

```
            oscA.volume.linearRampToValueAtTime(-6, Tone.now() + 0.06);
            oscB.volume.linearRampToValueAtTime(-Infinity, Tone.now() + 0.06);
        }
};
```

En aquest segon exemple, per assignar l'emmudiment selectiu dels dos components de l'oscil·lador de quadratura (només canal x, només canal y, ambdós canals x i y), el gestor d'events de la GUI crida la funció enviant-li tant el valor llegit de l'element de la GUI (en aquest cas, el "*select*" amb els tres valors pre-assignats) com els oscil·ladors sobre els que ha d'aplicar aquest valor. D'igual manera que en el fragment anterior, el canvi de valors no és instantani sinó gradual durant un període de temps de 60 mili-segons.

10.3.3 Implementació de la matriu de modulació

La matriu de modulació consisteix en un nombre arbitrari de fonts de senyal que envien una quantitat de senyal determinada a una o vàries destinacions alhora.

Les fonts de senyal disponibles són cadascun dels oscil·ladors, i més concretament, el seu component sinusoïdal. Les destinacions són les entrades de modulació de freqüència i d'amplitud de cadascun dels oscil·ladors i, addicionalment, la sortida mestra de volum i l'enviament (en mode "post-fader") a l'efecte de reverberació.

```
qm.engine.modulators = {
    osc1ToOsc3FMamt: qm.engine.nodes.osc1a.send("osc3FreqChan", -Infinity)
}


qm.engine.busses = function () {
    qm.engine.nodes.osc3a.frequency.receive("osc3FreqChan");
    qm.engine.nodes.osc3b.frequency.receive("osc3FreqChan");
}
```

En el primer fragment de codi anterior podem observar com el node la matriu de modulació, `qm.engine.modulators.osc1ToOsc3FMamt` és un Tone.Gain que controla la quantitat de senyal que enviem desde l'oscil·lador 1 cap al bus `osc3FreqChan` i es defineix amb el mètode `.send("oscFreqChan", -Infinity)`. D'altra banda, en el segon fragment de codi, podem observar com el que assigna la sortida del bus esmentat a l'entrada de modulació de freqüència dels dos sub-oscil·ladors de l'oscil·lador 3 és el mètode `.receive("osc3FreqChan")`.

Aquestes assignacions es repeteixen per cadascuna de les sortides i per cadascuna de les entrades, el que ens dóna un total de 4 sortides * 8 entrades = 32 nodes de la matriu de modulació de freqüència i amplitud; més 4 nodes addicionals per la sortida cap a la sortida mestra i 4 nodes cap al bus de reverberació. En total, la matriu de modulació consta de 40 mòduls disposats en 4 files de fonts de modulació i 10 columnes de sortides de modulació.

El següent fragment de codi assigna la funció a executar quan el gestor d'events de NexusUI detecta que hem actuat sobre l'element de la GUI corresponent a la quantitat de senyal a enviar des de l'oscil·lador 1 fins a l'entrada de modulació de freqüència de l'oscil·lador 3. Aquesta crida només envia el value corresponent a la posició del dial quan es fa per part de NexusUI.

```
qm.ui.binds = function () {
  qm.ui.e.dials.dialOsc1ToOsc3Freq.on('change', qm.ui.changes.dialOsc1ToOsc3FM);
}
```

La funció a executar quan s'interactua amb un element de la GUI és la que segueix a continuació. Podem observar que pot rebre, a més de *v*, un dial sencer (amb el seu corresponent `value`). Això es fa per diferenciar la crida quan es fa per part del gestor de presets de quan es fa per haver interactuat amb la GUI, i així evitar un bucle infinit resultant d'assignar un valor directament a l'element de la GUI, que automàticament tornaria a cridar la gestió del seu event corresponent, i així, fins a l'infinit.

```
qm.ui.changes = {
    dialOsc1ToOsc1FM: function(v, dial) {
        if(dial) dial.value = v;
        qm.ui.assignaModAmt(qm.engine.modulators.osc1ToOsc1FMamt, 20, v);
    },
}
```

La funció `qm.ui.assignaModAmt()` rep com a paràmetres l'oscil·lador el `Tone.Gain` sobre el que actuarem per regular la quantitat de modulació, el *"threshold"* o llindar a partir del qual convertirem el valor a `-Infinity`, i el valor llegit per la GUI o pel gestor de presets.

```
qm.ui.assignaModAmt = function(tone_gain, th, v) {
        if (v == th) {
            tone_gain.gain.value = -Infinity;
        } else {
            tone_gain.gain.value = v;
        }
    }
```

10.3.4 Instal·lació inicial de Node.js i del framework Express.js

En aquesta etapa del procés de desenvolupament, el primer que cal fer és instal·lar Node.js. D'entre les diverses possibilitats, es va escollir descarregar l'instal·lador des de l'enllaç https://nodejs.org. Per comprovar que Node.js s'ha instal·lat correctament, es pot obrir la línia de comandaments (Terminal en OSX) i executar la següent instrucció:

```
$ node -v
```

A continuació, per instal·lar Express.js (34) cal fer el directori de l'aplicació el directori actual

```
$ cd Desktop/Quadramat
```

Es fa servir el comandament npm per crear un fitxer package.json per la nostra aplicació

```
$ npm init
```

Aquest comandament ens demana una sèrie de dades sobre el nom i la versió de la nostra aplicació, així com el punt d'entrada a la seva execució.

Després ja es pot instal·lar Express al directori de l'aplicació i guardar-lo dins la llista de dependències

```
$ npm install express --save
```

A continuació, es va fer servir l'eina generadora d'Express, express-generator (35), per crear ràpidament l'esquelet de l'aplicació. En primer lloc cal instal·lar la línia de comandaments d'Express.js

```
npm install express-generator -g
```

Podem visualitzar les opcions disponibles fent servir l'opció –h

```
express -h
```

I per crear l'app express amb el nom de QuadraMat dins la carpeta del mateix nom en el directori actual executarem el següent:

```
express QuadraMat
```

A continuació, per instal·lar les dependències de l'aplicació QuadraMat

```
cd QuadraMat
npm install
```

Per executar l'aplicació només cal executar

```
npm start
```

...i a continuació carregar la pàgina http://localhost:3000 en el navegador.

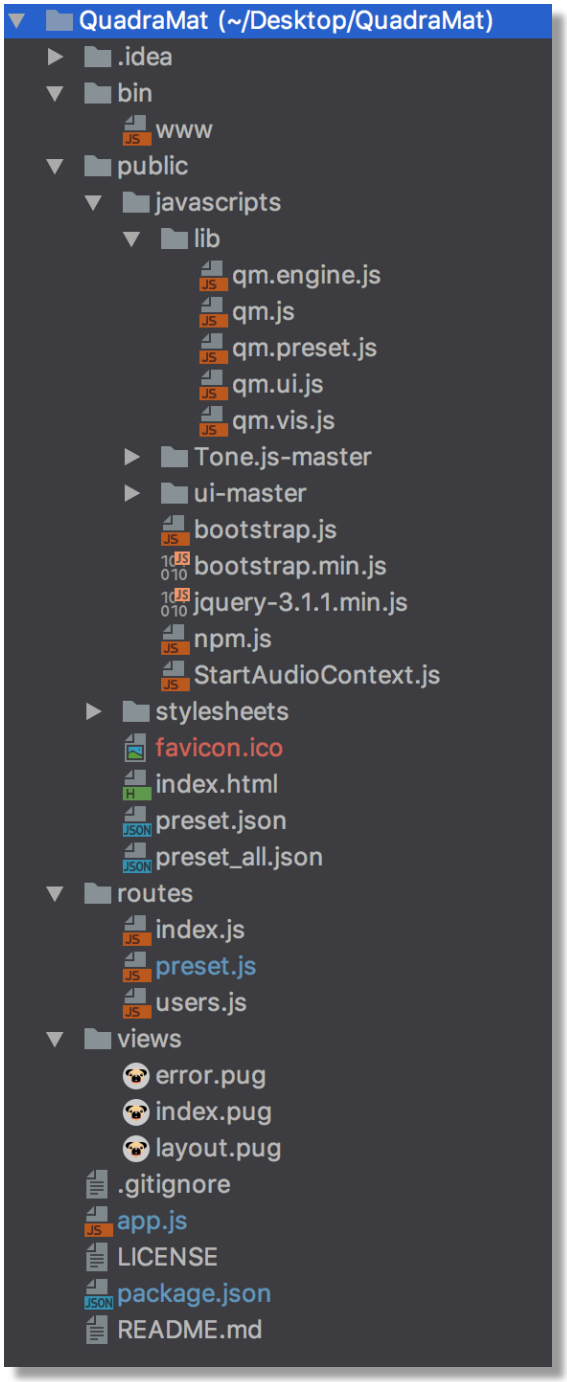
Si bé és cert que per facilitar les tasques de desenvolupament, també es va instal·lar la utilitat nodemon, que permet monitoritzar quan es canvia el codi i reiniciar el servidor automàticament. Així no cal tancar el servidor i tornar-lo a obrir cada vegada que es realitza un petit canvi en el seu codi. Per instal·lar nodemon es fa servir npm de la manera habitual

```
npm install -g nodemon
```

Finalment, per executar l'aplicació amb nodemon només cal desplaçar-se al directori de l'aplicació i teclejar el comandament

```
nodemon
```

10.3.5 Estructura de fitxers i configuracions del servidor



Il·lustració 11 - Estructura de fitxers del projecte

La carpeta `/bin` conté el codi que s'executarà quan es teclegi `node start`, segons està definit a Package.json. En executar node start, s'executa el codi a /bin/www, que inicialitza i arranca el servidor.

L'aplicació de servidor principal en Express.js és `app.js`. Quan l'usuari fa una petició al servidor, el codi a app.js s'executa, a més del especificat a les rutes corresponents.

La carpeta `routes` conté els routers amb el *middleware* que que s'executa (segons està prèviament definit a `app.js`) abans d'invocar les rutes demanades. La ruta '`/`' correspon a la pàgina principal de l'aplicació, la ruta '`/preset`' a la gestió de presets i la connexió amb la base de dades, i la ruta '`/users`' a la gestió dels usuaris.

La carpeta `javascripts` conté el codi, tant propi (dins `lib`) com de llibreries, que s'executarà en el client.

La carpeta `stylesheets` conté les fulles d'estil CSS

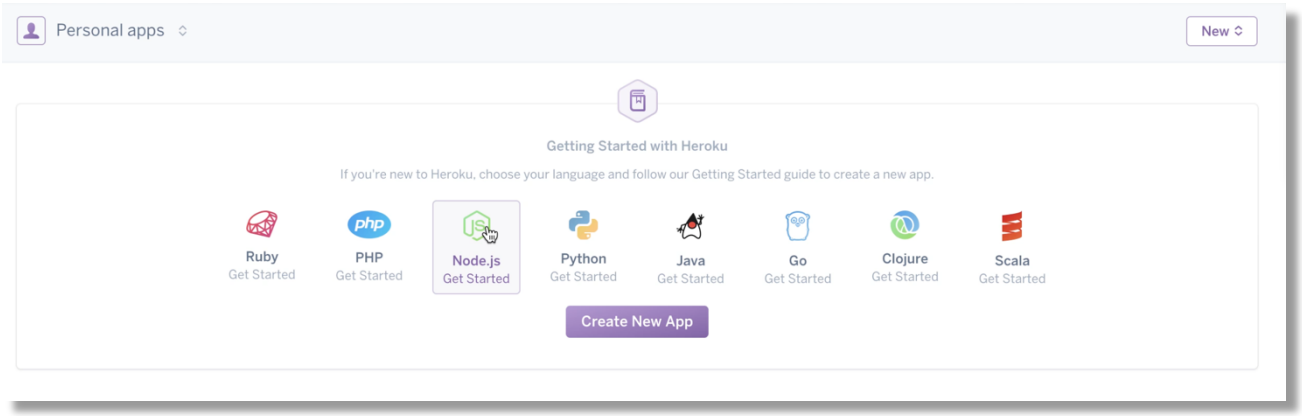
La carpeta `views` conté fitxers amb plantilles estàtiques que de moment no es fan servir.

El fitxer `package.json` conté informació sobre el nom i versió del programa, les dependències, localització dels binaris a executar, etc.

10.3.6 Desplegament al servidor heroku

El primer pas consisteix en registrar-se a la web de heroku.com. Ens demanaran algunes dades personals i un compte d'e-mail, que s'haurà de re-confirmar posteriorment, però es tracta d'un procés trivial.

Després de confirmar l'e-mail, la pàgina de benvinguda ja ens convida a escollir el llenguatge, que en el nostre cas és Node.js, i a crear una nova *app*.



Il·lustració 12 - Creació d'una nova app a Heroku

A partir d'aquí comença un procés que ens guiarà pas a pas pel procediment. Cal descarregar un client de Heroku que podrem fer servir des del Terminal del sistema operatiu. Aquest client es descarrega i instal·la de la manera habitual.

Una vegada instal·lat el client, des del Terminal, es tecleja `heroku login`, vinculant així la nostra consola al sistema de Heroku.

Entre els pre-requisits de la configuració local, figuren tenir instal·lat Node, npm i git, la versió dels quals podem comprovar des del nostre Terminal amb els comandaments `node –v`, `npm –v` i `git –version` respectivament.

Il·lustració 13 - Comprovació de la versió de Node i npm

La següent passa consisteix en preparar la *app*. La guia de Heroku ens suggereix clonar un repositori de mostra, però en el nostre cas es va escollir no fer-ho ja que l'aplicació Quadramat ja es troba llesta per ser pujada i, a més, ja disponible a GitHub, que s'ha fet

servir durant tot el procés de desenvolupament anterior. Així, el que hem de fer és desplaçar-nos a la carpeta on tenim instal·lada localment l'aplicació i, des d'allà, teclejar `heroku create` al Terminal. Aquest procés ens serveix per desplegar *("deploy")* l'aplicació a heroku. Aquest procés ens dóna com a resultat una adreça URL mitjançant la qual podrem accedir a la nostra aplicació.

```
Mac-Pro-de-Pere:desktop pere$ cd quadramat
Mac-Pro-de-Pere:quadramat pere$ ls
LICENSE         app.js          node_modules    public          views
README.md       bin             package.json    routes
Mac-Pro-de-Pere:quadramat pere$ heroku create
Creating app... done, ⬢ frozen-cliffs-27134
https://frozen-cliffs-27134.herokuapp.com/ | https://git.heroku.com/frozen-cliffs-27134.git
Mac-Pro-de-Pere:quadramat pere$
```

A partir d'aquest moment, quan es vulgui actualitzar el desplegament de l'aplicació en el servidor de heroku, només haurem de teclejar `git push heroku` i quan es vulgui actualitzar a GitHub, bastarà amb `git push`.

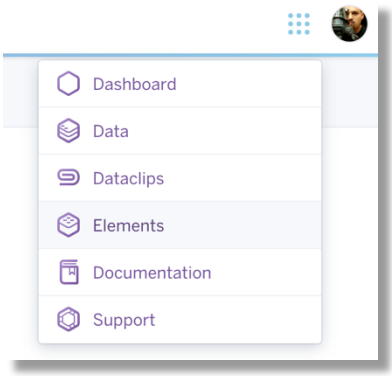
El procediment anterior proporciona una URL amb la que podem executar la nostra aplicació a heroku, que per defecte és https://frozen-cliffs-27134.herokuapp.com/.

Però per facilitar la memorització del nom de l'aplicació, es canvia el nom de l'aplicació dins el tauler de control de Heroku, escollint finalment el nom de http://quadramat.herokuapp.com/

10.4 Prototip #3: Sistema de presets i connexió amb back-end

10.4.1 Posta en servei de la base de dades

La base de dades la gestionarem des d'un proveïdor extern a Heroku en forma d' *add-on* i no directament mitjançant el propi servidor a Heroku. Això és degut a que Heroku no proporciona directament la possibilitat de fer servir MongoDB.

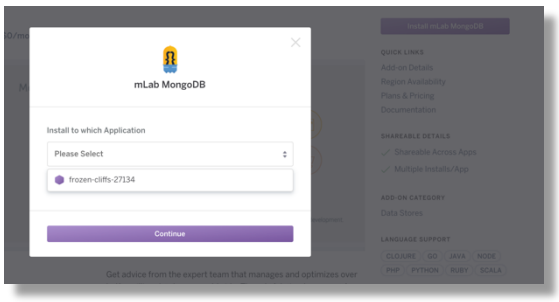


Il·lustració 14 - Menú principal de Heroku amb opció Elements

Però això no suposa cap problema ja que a l'apartat Elements del menú principal es pot trobar una tenda d'aplicacions coneguda com *Heroku Elements Marketplace*.

En aquest lloc web, dins l'apartat *Data Stores* podrem seleccionar mLAB MongoDB entre les moltes opcions disponibles.

mLab MongoDB ens ofereix un pla gratuït de tipus *sandbox* amb algunes limitacions però que cobreix sobradament les necessitats del projecte actual. Segons la informació disponible a la web de Heroku (36) *"mLab és un servei de base de dades en el núvol que disposa d'aprovisionament i escalament de bases de dades automatitzats, còpia de seguretat i recuperació, monitorització i alerta 24/7, eines de gestió basades en web i suport expert. La plataforma de tipus 'base de dades com a servei' de mLab proporciona centenars de milers de bases de dades a través de AWS, Azure i Google i permet als desenvolupadors centrar la seva atenció en el desenvolupament de productes en comptes de les operacions de gestió de sistemes. A través del complement mLab, els usuaris d'Heroku poden tenir a l'instant bases de dades de MongoDB en Amazon EC2 disponibles per a les seves aplicacions Heroku."*



Il·lustració 15 - Instal·lació de MongoDB des de Heroku

El procés d'instal·lació de mLab MongoDB dins Heroku és força fàcil ja que es realitza des del propi back-end de Heroku. Després de clicar sobre el botó d'instal·lació se'ns demana escollir l'aplicació de Heroku sobre la que volem instal·lar el servei. Dins el menú desplegable seleccionem frozen-cliffs-27134 (el nom que Heroku va donar per defecte a la nostra aplicació) i

Amb el servei de la base de dades ja instal·lat podem consultar a la pàgina del gestor

També és recomanable instal•lar el client de MongoDB al nostre ordinador. Al manual en línia (37) trobarem les instruccions necessàries per fer-ho. Una vegada instal•lat teclejam mongo al Terminal amb `mongo ds113775.mlab.com:13775/heroku_5062brkz -u <dbuser> -p <dbpassword>` substituïnt pel nostre nom d'usuari i password i entrarem a un shell des

del que podrem interactuar amb la base de dades. La pàgina d'ajuda oficial de MongoDB ens proporciona informació sobre el que podem fer des d'aquest shell (37).

Alguns exemples d'opcions de comandaments disponibles són:

```
db // fa referència a la base de dades actual
use database_name // per canviar a una altra base de dades
db.presets.insertOne({x:0}); // insereix document a la col·lecció presets
db.presets.deleteOne({x:0}); // esborra el primer document que compleix condició
```

10.4.2 Estructura del JSON d'un preset

Per començar amb les primeres proves del sistema de *presets* hem creat manualment un *preset* amb unes dades arbitràries, que ens servirà per inserir-lo a la base de dades i tenir algun contingut inicial amb el que poder treballar per implementar el selector de *presets*.

Com es pot observar, el preset presenta una estructura niada amb els elements superiors "id", "nom", "autor", "categoria", "osc1", "osc2", "osc3", "osc4" i "modmatrix". Els primers quatre elements emmagatzemen l'identificador únic del *preset* i algunes metadades mentre que els darrers cinc contenen els valors de tots els paràmetres del sintetitzador tal i com es mostren a la GUI.


```
{
  "id": 1,
  "nom": "Preset1",
  "rating": 0,
  "autor": {
    "nom": "Pere",
    "email": "pereamengual@gmail.com"
  },
  "categoria": "techno",
  "osc1": {
    "oct": 5,
    "semi": 0,
    "cents": 0,
    "wave": "sine",
    "xy": "xy",
    "order": 0,
    "even": true,
    "index": 0
  },
  "osc2": {
    "oct": 0,
    "semi": 0,
    "cents": 0,
    "wave": "sine",
    "xy": "xy",
    "order": 0,
    "even": true,
    "index": 0
  },
  "osc3": {
    "oct": 0,
    "semi": 0,
    "cents": 0,
    "wave": "sine",
    "xy": "xy",
```

```
    "order": 0,
    "even": true,
    "index": 0
  },
  "osc4": {
    "oct": 0,
    "semi": 0,
    "cents": 0,
    "wave": "sine",
    "xy": "xy",
    "order": 0,
    "even": true,
    "index": 0
  },
  "modmatrix": {
    "osc1dest": [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
    "osc2dest": [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
    "osc3dest": [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
    "osc4dest": [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ]
  }
},
```


10.4.3 Driver de MongoDB per Node.js

Segons la definició a la web de mongodb (38) *"Els controladors de MongoDB són les biblioteques de client que gestionen la interfície entre l'aplicació i els servidors i implementacions de MongoDB. Els controladors són responsables de gestionar connexions a instàncies independents de MongoDB i proporcionen els mètodes i interfícies que les aplicacions utilitzen per interactuar amb MongoDB".*

La primera passa per instal·lar aquest controlador és fer servir el gestor de paquets npm de la manera habitual

```
npm install mongodb --save
```

Per connectar amb MongoDB dins preset.js hi posarem el codi corresponent, que hem trobat a la pàgina corresponent de la documentació oficial de MongoDB (39).

```
var mongo = require('mongodb').MongoClient;
var assert = require('assert');
var url = 'mongodb://user:password@ds113775.mlab.com:13775/heroku_5062brkz';
mongo.connect(url, function(err, db) {
    assert.equal(null, err);
    console.log("Connected successfully to mongodb server.");
    var presetsCol = db.collection('presets');
        presetsCol.find().toArray(function(err, presets) {
        res.json(presets);
    })
        db.close();
});
```

10.4.4 Canvi de presets

Per poder canviar de *preset* amb el *select* de la GUI s'ha establert un *listener* amb el següent codi, que executa la funció descrita posteriorment:

```
qm.preset.domSelect.change(qm.preset.changePreset);


qm.preset.changePreset = function(event) {
    console.log($(this).val());
    var id = $(this).val();
    // Trobar el preset dins qm.preset.all
    var trobat = false;
    var i = 0;
    while(!trobat) {
        if(qm.preset.all[i].id == id) {
            trobat = true;
        } else {
            i++;
        }
    }
```

Per obtenir el conjunt dels *presets* de la base de dades es contacta amb la ruta `preset/all` via GET amb `$.get()`. El resultat queda emmagatzemat a data i a `qm.preset.all` (ja que el farà servir la funció `qm.preset.changePreset()`.

```
$.get('preset/all', function(data) {
    qm.preset.all = data;
    // Un cop tenim el JSON de tots els presets, hem d'omplir el selector HTML
    var html = '';
    for(var i = 0; i < data.length; i++) {
        // <option value="1">Preset1</option>
        html += '<option value="' + data[i].id + '">' + data[i].nom +
'</option>';
    }
    //modificam l'html del select amb
    qm.preset.domSelect.html(html);
});
```

10.4.x Procés de creació d'una icona de l'aplicació

En primer lloc es va realitzar una captura de pantalla d'una visualització d'una forma d'ona Lissajous proporcionada pel programa. A continuació, es va editar aquesta captura de pantalla amb un editor d'imatges, deixant el fons transparent i amb una eina en línia (40) es va crear la versió de la imatge amb les mides adequades i es va traslladar a la carpeta `/públic` del projecte.

A `app.js` es van haver de crear les següents línies de codi:

```
var favicon = require('express-favicon');
app.use(favicon(__dirname + '/public/favicon.png'));
```

10.5 Versió definitiva

10.5.1 Correcció de problemes de seguretat amb l'accés a la base de dades

Per evitar que la clau d'accés a la base de dades quedi visible al públic des del repositori a GitHub, es va incloure el valor de la variable `mongoUrl` dins un fitxer separat anomenat `config.js` que quedarà només disponible localment. Aquest fet s'ha documentat al fitxer `Readme.md` a manera de recordatori.

A més, a Heroku s'ha situat aquest valor com una variable de configuració, amb el codi corresponent per poder distingir on s'executa el codi.

```
var config = {mongoUrl: 'mongodb://userName:password@address'};
```

10.5.2 Implementació de la funcionalitat per emmagatzemar un preset

En primer lloc, es va assignar l'id `saveButton` al botó "Save" a `index.html` per poder accedir-lo assignarem l'element corresponent a la variable `qm.preset.saveButton` dins `qm.preset.js`. També es crea una funció per mostrar o amagar els diferents botons depenent de si l'usuari actual és el propietari del *preset*. Així, només el propietari actual pot guardar, reanomenar o esborrar un *preset*.

```
qm.preset.setButtonStates = function(preset) {
    qm.preset.domSaveButton.hide();
    qm.preset.domRenameButton.hide();
    qm.preset.domDeleteButton.hide();
    if('email' in preset.autor && 'email' in window.sessionStorage) {
        if(preset.autor.email === window.sessionStorage.email) {
            qm.preset.domSaveButton.show();
            qm.preset.domRenameButton.show();
            qm.preset.domDeleteButton.show();
        }
    }
}
```

A continuació, es varen definir les següents funcions per implementar les accions bàsiques de la gestió de *presets*: canvi de preset/lectura i creació/enregistrament. També es varen implementar algunes funcions auxiliars comunes a les dues accions principals per refrescar la llista de *presets*.

```
// Canvia de preset
qm.preset.changePreset = function(event) {...};

// Crea un objecte 'preset' en clicar sobre el botó Save
// si passem un nom vol dir que volem renombrar ("save as...")
// si preserveID és true vol dir que volem fer Update ("save" o "rename")
qm.preset.buildPresetObject = function(nom, preserveID) {...};

// Obté l'id d'un preset a partir del seu nom i l'email del seu autor
qm.preset.getPresetIdByNameAndEmail= function(nom, email) {...};

// Crida la ruta preset/update via POST i actualitza preset amb valors actuals
qm.preset.savePreset = function(preset) {...};

// Crida la ruta preset/save via POST enviant el nou preset
qm.preset.saveNewPreset = function(event) {...};
```

```
// Crida la ruta preset/update via POST i canvia el nom del preset
qm.preset.renamePreset = function(event) {...};

// Crida la ruta preset/delete via POST i elimina el preset de la base de dades
qm.preset.deletePreset = function(event)


// Refresca l'element HTML 'select2' a index.html amb la llista de presets
qm.preset.refreshPresetSelector = function() {...};

// Crida la ruta preset/all via GET i obtenir la llista de presets
qm.preset.getPresets = function() {...};

// Inicialització d'aquest mòdul
qm.preset.init = function() {...}
```

`qm.preset.all` és un *array* que conté el conjunt de *presets*. Ja que les mides dels *presets* són relativament petites (menys de 4 kB per preset) es va optar per carregar tota la base de dades i treballar-hi amb el seu conjunt, en lloc de carregar només les dades necessàries per emplenar el selector i carregar de la base de dades només el *preset* actual. A efectes pràctics, el rendiment de les dues alternatives és gairebé el mateix.

Per canviar de *preset*, en primer lloc, l'event creat en fer servir el selector de *presets* (un element select a `index.html`) crida la funció `qm.preset.changePreset()`, que fa servir el valor rebut per cercar el *preset* dins l'*array* esmentat anteriorment. A continuació, es fan servir els valors del *preset* obtingut per canviar els paràmetres de la GUI fent servir les funcions `qm.ui.changes.xxx` definides a qm.ui.js i d'aquesta manera assegurar que s'executa correctament el codi corresponent a les interaccions entre els diferents elements (com per exemple, el càlcul de la freqüència a partir del valor de tres botons). Pels elements de la GUI com *selects* i *toggles* es modifica directament el seu valor fent servir les variables `qm.ui.e.selects.xxx.value` i `qm.ui.e.toggles.xxx.state`, ja que no interactuen amb cap altre element de la GUI.

10.5.3 Selector millorat per carregar presets i botons prev/next

Per implementar el selector s'ha fet servir la llibreria select2 ja que presenta millores substancials en usabilitat respecte de l'element `<select>` bàsic proporcionat per HTML. Entre aquestes millores destaquem la possibilitat de realitzar cerques tant per nom d'usuari com per nom de *preset* amb detalls com la cerca amb accents diacrítics, desplaçament infinit, compatibilitat amb gairebé tots els navegadors i un aspecte personalitzable mitjançant CSS.

En primer lloc, tant a index.html com a qm.preset.js s'ha d'incloure el codi dels marcadors com la seva corresponent assignació a variables i els seus *listeners*.

```
qm.preset.prevButton = $('#prev');
qm.preset.nextButton = $('#next');
qm.preset.domSelect2 = $('#presets2');


qm.preset.prevButton.click(qm.preset.prevPreset);
qm.preset.nextButton.click(qm.preset.nextPreset);
qm.preset.domSelect2.on('change', qm.preset.changePreset);
```

Addicionalment, per emmagatzemar el valor del nombre del *preset* dins la llista actual i poder realitzar canvis al *preset* següent i anterior, s'emmagatzema aquesta informació dins la variable qm.preset.cursor. Dues funcions s'executen per passar al *preset* anterior i posterior, respectivament.

```
qm.preset.prevPreset = function(){...};
qm.preset.nextPreset = function(){...};
```

El codi a preset.js per la ruta `preset/all` quan es crida amb GET ens retorna la llista de *presets* ordenada per autor i en format JSON.

```
router.get('/all', function(req, res, next) {
    mongo.connect(config.mongoUrl, function(err, db) {
        assert.equal(null, err);
        console.log("Connected successfully to mongodb server.");

        var presetsCol = db.collection('presets');

        presetsCol.find().sort({"autor.nom" : 1, "nom":
1}).toArray(function(err, presets) {
            res.json(presets);
            db.close();
        });

    });
});
```

Per guardar un nou *preset* a la base de dades, s'accedeix a la ruta `preset/save` via POST enviant el *preset* actual en format JSON. Primer es construeix l'objecte *preset* amb la funció auxiliar construïda expressament amb aquesta finalitat. A més, posem el nom del *preset* actual al selector com indicació de que el procés s'ha efectuat correctament.

```
qm.preset.saveNewPreset = function(event) {
    var nom = prompt("Posa un nom al preset", "");
    if(nom != null) {
        var preset = qm.preset.buildPresetObject(nom, false);
        $.ajax({
            url: 'preset/save',
            type: 'POST',
            contentType: "application/json",
            data: JSON.stringify(preset),
            success: function (data) {
                qm.preset.all = data;
                qm.preset.refreshPresetSelector();
                var _id = qm.preset.getPresetIdByNameAndEmail(nom,
preset.autor.email);
                qm.preset.domSelect2.val(_id);
                qm.preset.domSelect2.trigger('change'); },
            error: function (err) {
                console.error(err);
            }
        });
    }
};
```

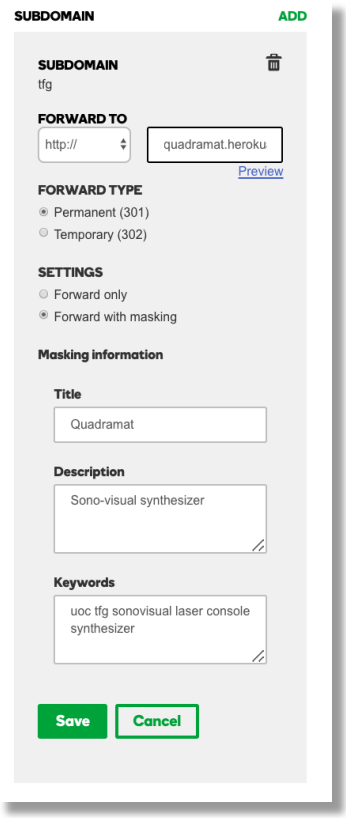
A més, en el servidor també s'ha implementat la funcionalitat corresponent al botó "save", que en realitat el que fa és actualitzar dins la base de dades el *preset* corresponent al id del *preset* actual. Podem observar com l'operador `$set` reemplaça el valor d'un camp amb el valor especificat, en el nostre cas, el `req.body` corresponent als valors dels paràmetres del *preset*, segons han estat rebuts a la ruta `preset/update` via POST.

```
presetsCol.updateOne({ _id: ObjectID(_id) }, { $set: req.body }, function(err,
result) {
    assert.equal(null,  err);
    presetsCol.find().toArray(function(err, presets) {
        assert.equal(null, err);
        db.close();
        res.json(presets);
    });
});
```

Per esborrar un preset mitjançant l'ús del botó "delete" a la GUI, es crida la ruta `preset/delete` via POST i s'esborra l'entrada sencera a la base de dades fent referència a l'identificador únic del *preset*.

```
presetsCol.deleteOne({ _id: ObjectID(_id) }, function(err, result) {
    assert.equal(null, err);
    presetsCol.find().toArray(function(err, presets) {
        assert.equal(null, err);
        db.close();
        res.json(presets);
    });
});
```

10.5.4 Canvi de nom de l'aplicació i enllaç a un subdomini propi



Il·lustració 16 - Tauler de control de redireccionament

La memorabilitat (41) és un dels aspectes que contribueixen positivament a la usabilitat d'un lloc web. Encara que l'aplicació d'aquest principi està més relacionada amb que es puguin recordar els passos per realitzar els procediments dins un lloc web, no està de més tenir en compte que el nom del lloc web és, en sí mateix, la principal clau d'entrada per visitar-lo. No és estrany que un usuari no pugui accedir a un lloc web perquè n'ha oblidat el nom de la seva adreça (o URI) i, amb aquests precedents l'adreça de l'aplicació web s'ha traslladat a un subdomini de graumultimedia.com.

A més, i només a efectes de desenvolupament, també s'ha canviat l'adreça on es pot accedir directament a l'aplicació a Heroku, donant-li un nom més fàcil de recordar.

El procediment per redireccionar l'aplicació a un subdomini és senzill i només requereix entrar al tauler de control de gestió del servidor i afegir el redireccionament especificant el nom del subdomini, l'adreça a la que redirigirà, el tipus (permanent o temporal) i els ajustaments d'emmascarament.

Inicialment el projecte va fer servir un redireccionament amb masking, que oculta a l'usuari l'URI real de l'aplicació. No obstant, en els darrers dies de desenvolupament aquesta opció ha deixat de funcionar correctament i es fa servir un redireccionament sense màscara, que mostra al navegador l'URI de l'aplicació a Heroku.

La adreça definitiva de l'aplicació és http://tfg.graumultimedia.com/

10.5.5 Sistema de gestió d'usuaris

Aquesta part ha resultat molt més complexa del que s'havia previst inicialment. Tot i així, s'ha implementat un sistema de gestió d'usuaris que els permet emmagatzemar i compartir els seus propis *presets*, així com accedir als *presets* dels altres usuaris.

Els usuaris s'identifiquen de forma única per la seva adreça de correu electrònic i, addicionalment, es donen a conèixer a la resta d'usuaris mitjançant el seu nom d'usuari o *nickname*.

Arribarem a la ruta `register/` via GET mitjançant l'enllaç del botó corresponent a la pàgina principal de l'aplicació. En aquesta ruta arrel es serveix la pàgina dinàmica `/register.pug` que és la que ens proporcionarà el formulari corresponent. Altres rutes són `register/welcome` via GET que ens porta a `/welcome.pug` amb una pàgina de benvinguda i `register/new` via POST que conté el codi per crear un nou usuari. D'aquest darrer codi destaquem la detecció prèvia d'un usuari amb mateix nom o email que retorna un error que farà servir la pàgina encarregada de proporcionar el contingut a l'usuari per informar-lo.

10.5.6 Gestió de sessions d'usuari

Amb `express-session` (42) identifiquem l'usuari de la sessió actual mitjançant la correspondència entre una *cookie* emmagatzemada localment a l'ordinador de l'usuari i una entrada a una base de dades específica dins el servidor implementada com la col·lecció "users".

`Express-session` ens permet emmagatzemar informació per cada usuari que visita la web, i així fer possible que es puguin realitzar les accions bàsiques de gestió dels *presets* (llegir, guardar, etc.) sense haver d'iniciar la sessió constantment.

Per assegurar la persistència de les dades fins i tot després d'haver reiniciat l'aplicació al servidor, en comptes de fer servir `MemoryStore` es fa servir `MongoDBStore`.

Per assegurar la privacitat dels usuaris cal encriptar les dades de la sessió, i això es fa mitjançant bcrypt (43) que permet fer un *hash* dels passwords i emmagatzemarlos de forma segura. Es fa servir un valor de `10` pel paràmetre `salt` del bcrypt per obtenir un nivell d'encriptació suficient.

```
bcrypt.hash(req.body.password, 10, function(err, hash) {
    // Store hash in database
    user.password = hash;
    // {...}
}
```

10.5.7 Canvi de pàgines estàtiques a pàgines dinàmiques al servidor.

Fins la fase actual del projecte, es modificaven només puntualment alguns elements HTML i les seves propietats mitjançant la tècnica coneguda sota les sigles AJAX (Asyncronous Javascript and XML). Però a causa de la creixent complexitat dels continguts dinàmics de l'aplicació es decideix fer servir el *template engine* Pug (abans Jade) i servir els continguts HTML de forma dinàmica. Així, les anteriors pàgines HTML s'han de convertir a Pug i, com aquest pot resultar una mica tediós i repetitiu, es fa servir el convertidor en línia a http://html2jade.org/.

Dins la carpeta `views` ara es troben les següents pàgines: `debug.pug`, `error.pug`, `index.pug`, `layout.pug`, `login.pug`, `register.pug` i `welcome.pug`. Es poden veure exemples comentats del codi corresponent a l'annex 2.3 del present document.

10.5.8 Codis d'invitació

Per evitar que usuaris desconeguts facin un mal ús de l'aplicació, inicialment només es permetrà donar-se d'alta com usuari amb un codi d'invitació. No obstant, els visitants podran obtenir els presets creats pels altres usuaris i modificar els seus paràmetres, encara que no els podran guardar ni, evidentment, renombrar o esborrar.

El codi d'invitació és: LISSAJOUS

11. APIs i llibreries utilitzades

11.1 Web Audio API

La Api Web Audio (44) és una especificació desenvolupada pel *World Wide Web Consortium* (W3C) i descriu una interfície de programació d'aplicacions per processar i sintetitzar àudio en aplicacions web. El paradigma principal és un graf d'enrutament d'àudio, on es connecten diversos objectes *AudioNode* per definir la representació d'àudio en general. El processament real es porta a terme principalment en la implementació subjacent (típicament ensamblador optimitzat, C o C + +), però també es pot utilitzar el processament i la síntesi de JavaScript mitjançant *AudioProcessorNode* o, en properes implementacions, *AudioWorkers*.

La API de Web Audio ens proporciona una varietat d'interfícies i events associats que podem dividir en les següents categories (45):

11.1.1 Definició del grafs general d'àudio

Contenidors generals i definicions que donen forma als grafs d'àudio com `AudioContext`, `AudioNode` i `AudioParam`. `OfflineAudioContext`, en canvi, no renderitza l'àudio sinó que el genera, el més ràpid possible, en un buffer.

11.1.2 Definició de fonts d'àudio

Interfícies que defineixen les fonts d'àudio a fer servir en l'API Web Audio, com `OscillatorNode`, `AudioBuffer`, `AudioBufferSourceNode`, `MediaElementAudioSourceNode` o `MediaStreamAudioSourceNode`

11.1.3 Definició de filtres i efectes d'àudio

Interfícies que defineixen efectes i processat que es pot aplicar a les fons d'àudio, com `BiquadFilterNode`, `ConvolverNode`, `DelayNode`, `DynamicsCompressorNode`, `GainNode`, `WaveshaperNode`, `IIRFilterNode`, etc.

11.1.4 Definició de destinacions d'àudio

Interfícies que defineixen on es pot enviar l'àudio final com `AudioDestinationNode` i `MediaStreamAudioDestinationNode`

11.1.5 Anàlisi de dades i visualització

Permet extreure dades de l'àudio tant en els dominis de temps com de freqüència mitjançant `AnalyserNode`

11.1.6 Divisió i fusió de canals

Per dividir i fusionar canals es fan servir les interfícies `ChannelSplitterNode` i `ChannelMergeNode`

11.1.7 Espacialització d'àudio

Interfícies que permeten situar una font sonora dins un espai virtual com `AudioListener`, `PannerNode` i `StereoPannerNode`.

11.1.8 Processat d'àudio en JavaScript

Encara que ja considerat obsolet en favor de les interfícies `AudioWorkers`, Web Audio permet processar àudio mitjançant codi JavaScript i la interfície `AudioScriptNode`.

11.2 Tone.js

Tone.js és un marc per crear música interactiva al navegador. Proporciona capacitats de programació avançades, sintetitzadors i efectes, i abstraccions musicals intuïtives construïdes a sobre de l'API d'àudio web.

En comparació a la llista d'interfícies i categories descrites en l'apartat anterior, Tone.js facilita la creació sonora mitjançant la utilització d'abstraccions de més alt nivell similars a les que fan servir els músics i els dissenyadors sonors en entorns de producció sonora més tradicionals.

La documentació de Tone.js ens ofereix una divisió dels nodes en les següents categories:

11.2.1 Component

Nodes que processen àudio, generen senyals de control o ajuden a realitzar enrutaments addicionals. Alguns dels nodes que s'han fet servir en el projecte són: `Filter`, `Limiter`, `Merge` i `Waveform`.

11.2.2 Control

Nodes que serveixen per crear estructures de control semblants seqüenciadors i arpegiadors. No s'han fet servir en el projecte.

11.2.3 Core

Nodes que formen part del nucli bàsic de la llibreria i serveixen com blocs fonamentals. En el projecte s'han fet servir *`Tone`* (classe base de totes les altres classes), `AudioNode` (com classe base per totes les classes que produeixen àudio), `Master` i `Gain`.

11.2.4 Effect

Nodes que processen senyals d'àudio. En el projecte s'han fet servir `Chebyshev` i `Freeverb`.

11.2.5 Event

Nodes relacionats amb la planificació d'events amb una finalitat musical . No s'han fet servir en el projecte.

11.2.6 Instrument

Abstraccions d'alt nivell que inclouen combinacions de nodes adequades per la creació de sons combinant nodes que generen i processen so, senyals de control, etc. No s'han fet servir en el projecte.

11.2.7 Signal

Nodes que gestionar la generació i processat de senyals de control. No s'han fer servir de forma explícita en el projecte.

11.2.8 Source

Nodes que generen sons. En el projecte només s'ha fet servir `Oscillator`.

11.2.9 Type

Nodes que permeten gestionar la conversió de tipus. No s'han fet servir en el projecte.

11.3 Canvas API

La API Canvas (46) permet dibuixar, mitjançant Javascript, a l'element contenidor `<canvas>`. Després d'obtenir l'objecte corresponent al canvas, s'hi pot aplicar el mètode `getContext()` per obtenir un objecte sobre el que disposem de mètodes i propietats per dibuixar al canvas. L'objecte obtingut mitjançant `getContext("2d")` es pot fer servir per dibuixar línies, text, capses, cercles, etc.

Alguns dels mètodes i propietats que s'han fet servir en el visualitzador del projecte són els següents:

```
canvasCtx.globalAlpha
canvasCtx.strokeStyle
canvasCtx.fillStyle
canvasCtx.clearRect();
canvasCtx.fillRect();
canvasCtx.beginPath();
canvasCtx.lineJoin
canvasCtx.lineWidth
canvasCtx.lineTo();
canvasCtx.stroke();
```

11.4 StartAudioContext.js

`StartAudioContext` (47) és una senzilla llibreria JavaScript que inicia l'`AudioContext` de Web Audio seguint una acció explícita d'usuari.

Es fa servir per assegurar compatibilitat amb els dispositius iOS (48) ja que, segons s'indica a la documentació d'Apple, "A iOS, l'API d'àudio web requereix que els sons es desencadenin a partir d'una acció explícita de l'usuari, com ara un toc. Cridar `noteOn()` des d'un esdeveniment `onload` no reproduirà el so."

11.5 Bootstrap (amb dependència de JQuery)

Bootstrap (49) és un conjunt d'eines de codi obert per al desenvolupament amb HTML, CSS i JS. Permet prototipar ràpidament pagines web o crear una aplicació amb variables Sass i mixins, un sistema de reixeta responsive *("responsive grid")*, molts components i plugins prefixats pre extensos prefixats i potents connectors integrats a jQuery.

En el projecte es fa servir la funcionalitat de reixeta responsive que proporciona Bootstrap per facilitar la disposició dels elements en pantalles de diferents mides, sempre tenint en compte el principi de *"mobile first"*; es a dir, s'ha de desenvolupar pensat inicialment en els usuaris dels dispositius mòbils.

11.6 NexusUI

NexusUI (50) és una col·lecció d'interfícies HTML5 i funcions auxiliars de Javascript per ajudar a crear instruments d'àudio web al navegador i gestionar-ne els events associats. A més de les interfícies, ofereix uns quants mètodes d'ajuda per a l'afinació i el control de la temporització d'events. No proporciona cap capacitat de producció de so, ja que aquesta es realitza amb ToneJs.

La documentació de NexusUI ens ofereix una llista de les interfícies i funcions auxiliars disponibles:

11.5.1 Core i General

En aquest apartat figuren les interfícies bàsiques que ofereix la llibreria. D'aquestes categories en el projecte s'ha fet servir `Dial`, `Select` i `Toggle`.

11.5.2 Mobile

No s'han fet servir les funcions específiques per dispositius mòbils, com l'accés als acceleròmetres mitjançant `Tilt`.

11.5.3 Espacialization

No s'han fet servir les interfícies específiques per panoramització que proporciona la llibreria.

11.5.4 Visualization

Encara que les funcions de visualització funcionen perfectament, s'ha optat per programar un visualitzador mitjançant canvas que resulta més fàcil de personalitzar.

11.5.5 Tuning

Per l'algorisme de creació d'un valor de freqüència a partir dels valors d'octava, semitò en escala justa i centèsimes de semitò, s'han fet servir `Tune`, `Scale` i `Note`.

11.7 Select2

Select2 (51) és una llibreria que ofereix un quadre de selecció personalitzable amb suport per a cerques, etiquetatge, conjunts de dades remots, desplaçament infinit i un aspecte visual més elaborat.

Aquesta llibreria permet no només una cerca més ràpida dels presets (que potencialment podrien arribar a ser centenars) sinó també una manera visualment atractiva de separar-los per autor. A més, funcionalitat com el desplaçament infinit ens facilita la usabilitat en pantalles petites o de dispositius mòbils.

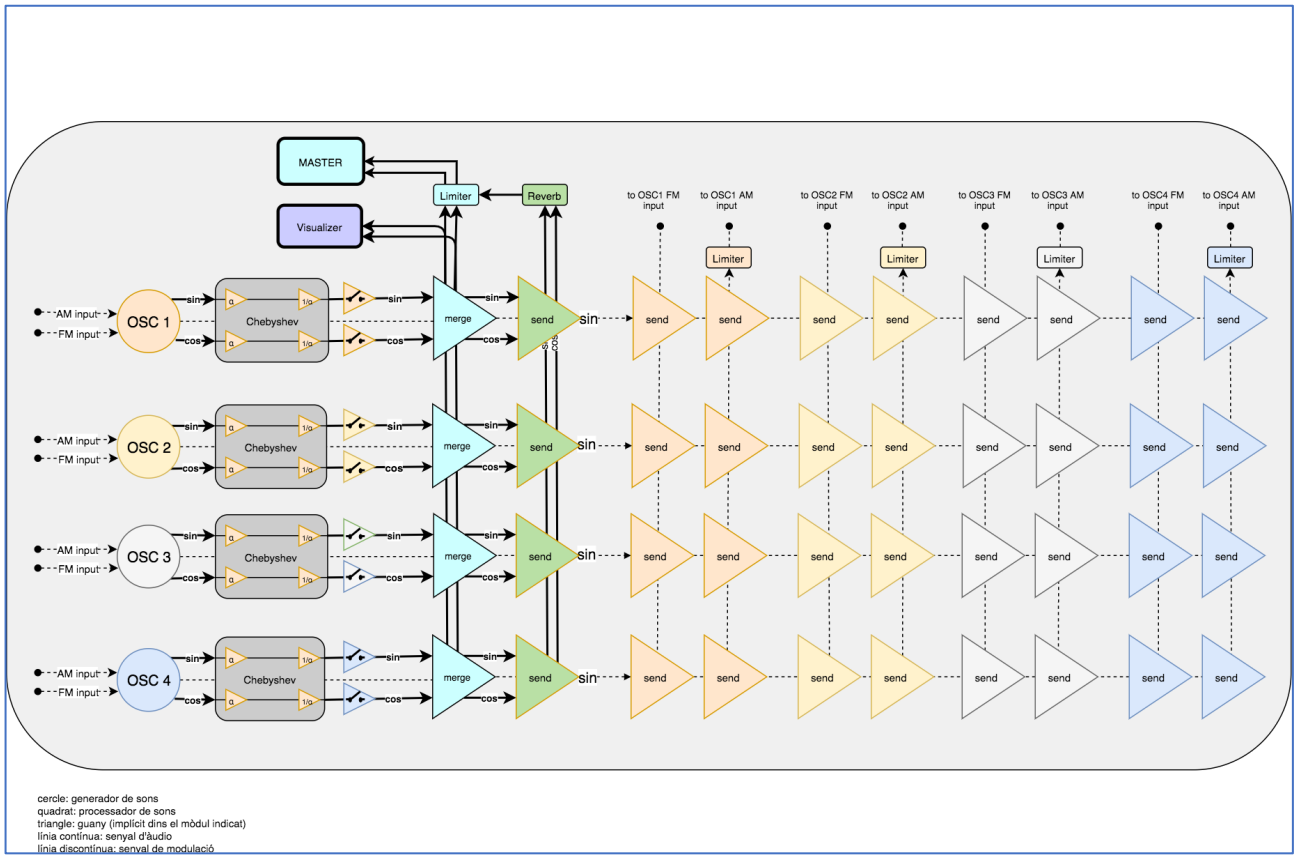
11.8 Llibreries i middleware de part del servidor

Un cop d'ull al fitxer `package.json` ens revela quins són els paquets que hem instal·lat i fet servir mitjançant npm, l'editor de paquets de Node:

- **Bcrypt** (43) per implementar l'encriptació de les contrasenyes amb una funció de hashing
- **Body-parser** (52) extreu tota la part del cos d'un flux de sol·licitud entrant i l'exposa com req-body
- **Connect-mongodb-session** (53) exporta una única funció que pren una instància de connexió i retorna una classe MongoDBStore que es pot utilitzar per emmagatzemar sessions en MongoDB.
- **Cookie-parser** (54) és un analitzador de galetes que analitza el encapçalament de la *cookie* i emplena req.cookies amb un objecte que conté els noms de les galetes.
- **Express.js** (34) és un marc d'aplicacions web Node.js mínim i flexible que proporciona un conjunt robust de característiques per a aplicacions web i mòbils.
- **Express-favicon** (55) és un *middleware* per servir la icona de la pàgina del navegador
- **Express-session** (56) és un *middleware* per gestionar les sessions en Express.js
- **Mongodb** (57) és el driver oficial de MongoDB per Node.js
- **Pug** (58) (abans conegut com Jade) és un motor de plantilles d'alt rendiment molt influït per Haml i implementat amb JavaScript per a Node.js i navegadors. Les plantilles es fan servir per servir contingut dinàmic als navegadors.

12. Diagrames

12.1 Diagrama de blocs del motor d'àudio (esbós final)



Il·lustració 17 – Diagrama de blocs del motor d'àudio #003

12.2 Descripció dels blocs

12.2.1 Generadors de sons (cercles)

- `Tone.Oscillator`: Oscil·lador de quadratura amb sortides desfasades 90 graus i entrades de modulació de freqüència (*thru-zero*) i amplitud (bipolar)

12.2.2 Processadors de sons (quadrats)

- `Tone.Chebyshev`
- `Tone.Limiter`
- `Tone.Freeverb`

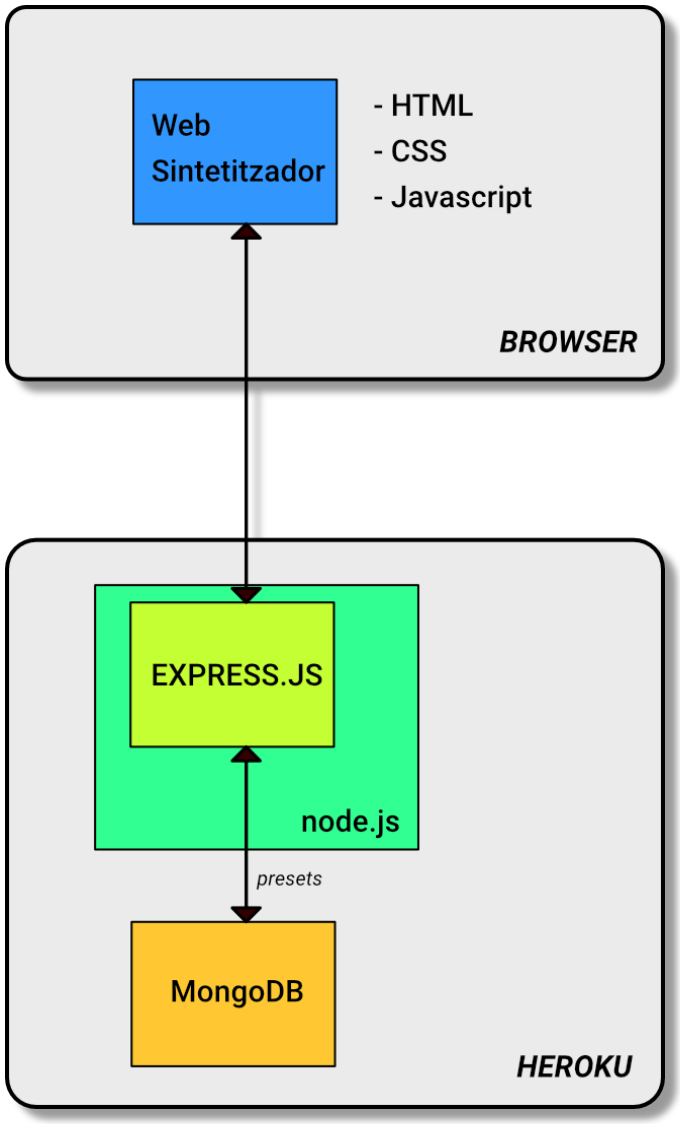
12.2.3 Controls de guany (triangles)

- Interruptors x/y/xy
- Enviaments a `Tone.Master`, `Tone.Reverb` i els bussos de modulació de freqüència i amplitud.

12.2.4 Destinacions finals (quadrats amb regruix)

- `Tone.Master`
- `Tone.Waveform`

12.3 Diagrama de l'arquitectura de l'aplicació



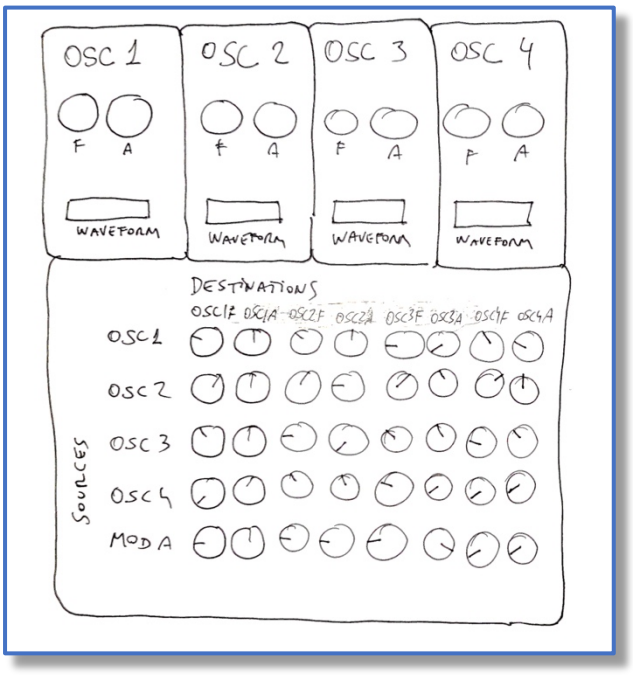
Il·lustració 18 - Diagrama de l'arquitectura de l'aplicació

13. Prototips de la interfície gràfica

13.1 Mockups Lo-Fi

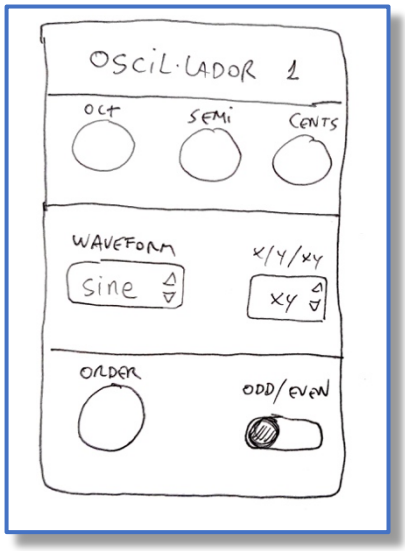
En la fase inicial del projecte es varen realitzar alguns *mockups* en baixa fidelitat, a ma alçada sobre paper, per esbrinar quina seria la disposició dels elements més adequada, així com per assistir en el procés de pluja d'idees.

13.1.1 Visió general de la pàgina principal

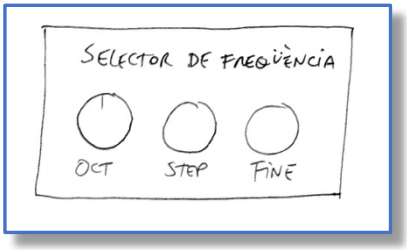


Il·lustració 19 - Visió general de la GUI

13.1.2 Detall de l'oscil·lador

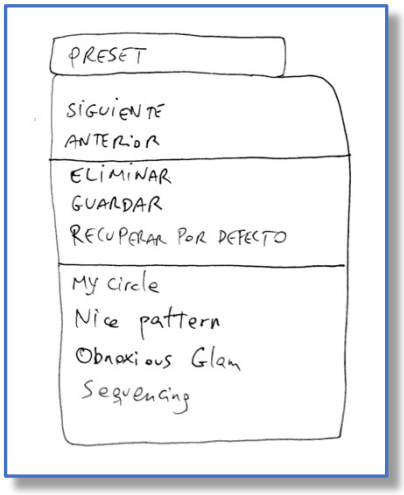


Il·lustració 20 - Detall de l'oscil·lador



Il·lustració 21 - Detall del selector de freqüència

13.1.5 Gestió de presets

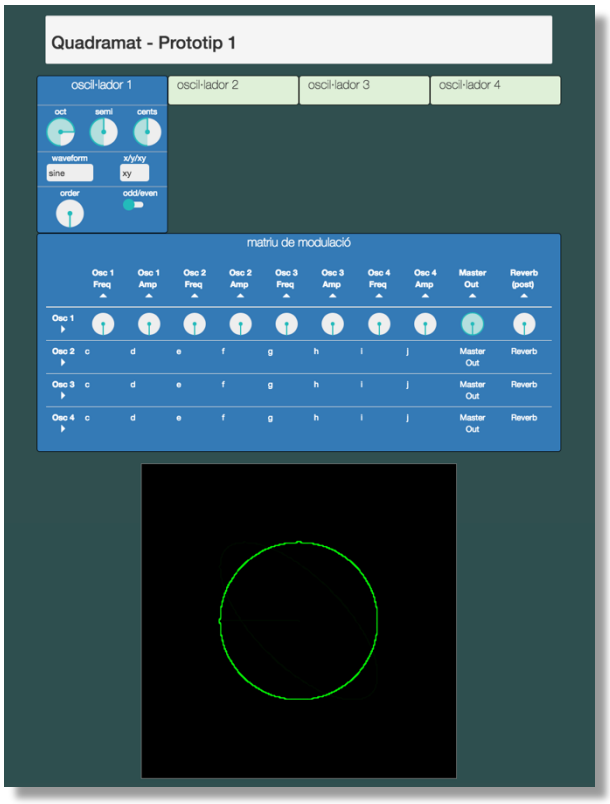


Il·lustració 22 - Detall del gestor de presets

13.4 Hi-Fi

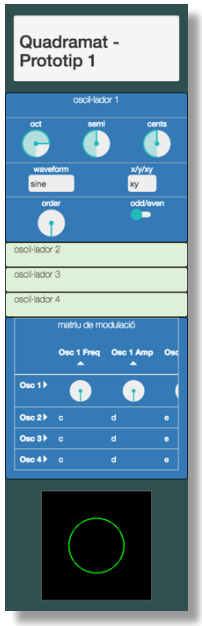
La metodologia de desenvolupament per prototips permet anar avançant també amb la mateixa filosofia pel que respecta a la interfície gràfica d'usuari que, en el cas del projecte que ens ocupa, no és de cap manera un afegit sinó una part essencial del producte final.

Així, el primer prototip ja disposa d'una interfície d'usuari raonablement acabada i que ens pot servir per il·lustrar la interfície gràfica que s'espera obtenir de l'aplicació en els seus estats finals. Com es pot comprovar, la interfície ja es *responsive* i adapta la disposició dels elements a les mides de la pantalla.



Il·lustració 25 - Laptop 1024px

Il·lustració 24 - Tablet 768px



Il·lustració 23 - Mobile small 320px

14. Perfils d'usuari

14.1 Perfils d'usuari

Per ajudar a definir els perfils d'usuari de l'aplicació, és aconsellable realitzar-ne una divisió en categories, segons la seva relació amb el producte

14.1.1 Primari

Es tracta d'un usuari que treballarà regularment o directament amb el producte: artistes visuals, músic experimentals, etc.

- Demografia: edat entre 16 i 80 anys, gènere indefinit, localització indefinida, estatus sòcio-econòmic mig, mig-alt.
- Posició actual i experiència: músic independent d'estil experimental, artista sonor, artista sono-visual. Experiència mínima de cinc anys.
- Informació de l'empresa o col·lectius de que forma part: membre de col·lectius de programació creativa, *maker*, i grups en xarxes socials relacionats amb les aplicacions artístiques de la investigació sonora i visual.
- Nivell educatiu: mig, mig-alt. Amb estudis de Música, Belles Arts i/o Programació.
- Experiència amb ordinadors: alta, molt alta.
- Experiència específica amb sintetitzadors i eines sono-visuals: alta, molt alta.
- Tasques que realitzarà: investigació sonora per la creació de preajustaments, generació en directe i amb públic de patrons sono-visuals.
- Tecnologia de que disposa: ordinador d'escriptori o portàtil, tauleta i mòbil.
- Actituds i valors: gaudeix del risc i els productes culturals més avançats, afavoreix l'ús de tecnologies d'avantguarda en el seu procés creatiu.

14.1.2 Secundari

Aquest tipus d'usuari fa servir el producte infreqüentment: interessats ocasionals en la programació sono-visual, curiosos per la tecnologia, etc.

- Demografia: edat entre 16 i 80 anys, gènere indefinit, localització indefinida, estatus sòcio-econòmic mig, mig-alt.
- Posició actual i experiència: entusiasta de les noves tecnologies, aficionat al art sonor i l'experimentació audiovisual.
- Informació de l'empresa o col·lectius de que forma part: membre de grups en xarxes socials relacionats amb les aplicacions artístiques de la investigació sonora i visual, estudiant de Belles Arts, estudiant de música.
- Nivell educatiu: mig, mig-alt. Amb estudis bàsic sobre música i/o creació visual.
- Experiència amb ordinadors: mitja, alta.

- Experiència específica amb sintetitzadors i eines sono-visuals: cap, baixa.
- Tasques que realitzarà: experimentació superficial de les possibilitats creatives del dispositiu.
- Tecnologia de que disposa: ordinador d'escriptori o portàtil, tauleta i mòbil.
- Actituds i valors: gaudeix del risc i els productes culturals més avançats, *"early-adopter"*.

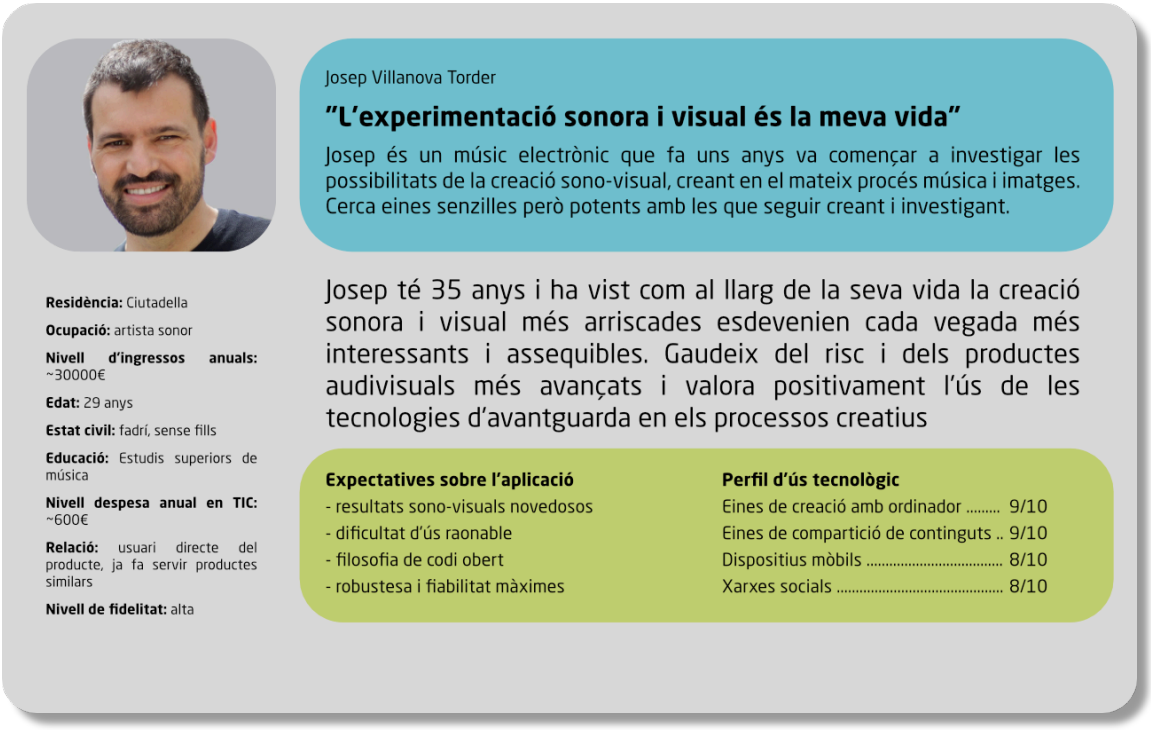
14.1.3 Terciari

Aquest usuaris es veuen afectats de forma indirecte pel producte o pels que l'usen: públic a events en directe, visionadors de vídeos en línia, etc.

- Demografia: edat entre 16 i 80 anys, gènere indefinit, localització indefinida, estatus sòcio-econòmic mig, mig-alt.
- Posició actual i experiència: interessat les noves tecnologies, interessat ocasionalment en la creació audiovisual contemporània.
- Informació de l'empresa o col·lectius de que forma part: membre de grups en xarxes socials generalistes.
- Nivell educatiu: mig, mig-alt.
- Experiència amb ordinadors: mitja, alta.
- Experiència específica amb sintetitzadors i eines sono-visuals: cap, molt baixa.
- Tasques que realitzarà: observador en qualitat de públic dels resultats de les creacions sono-visuals d'altres persones.
- Tecnologia de que disposa: tauleta o mòbil.
- Actituds i valors: curiositat pels esdeveniments musicals contemporanis.

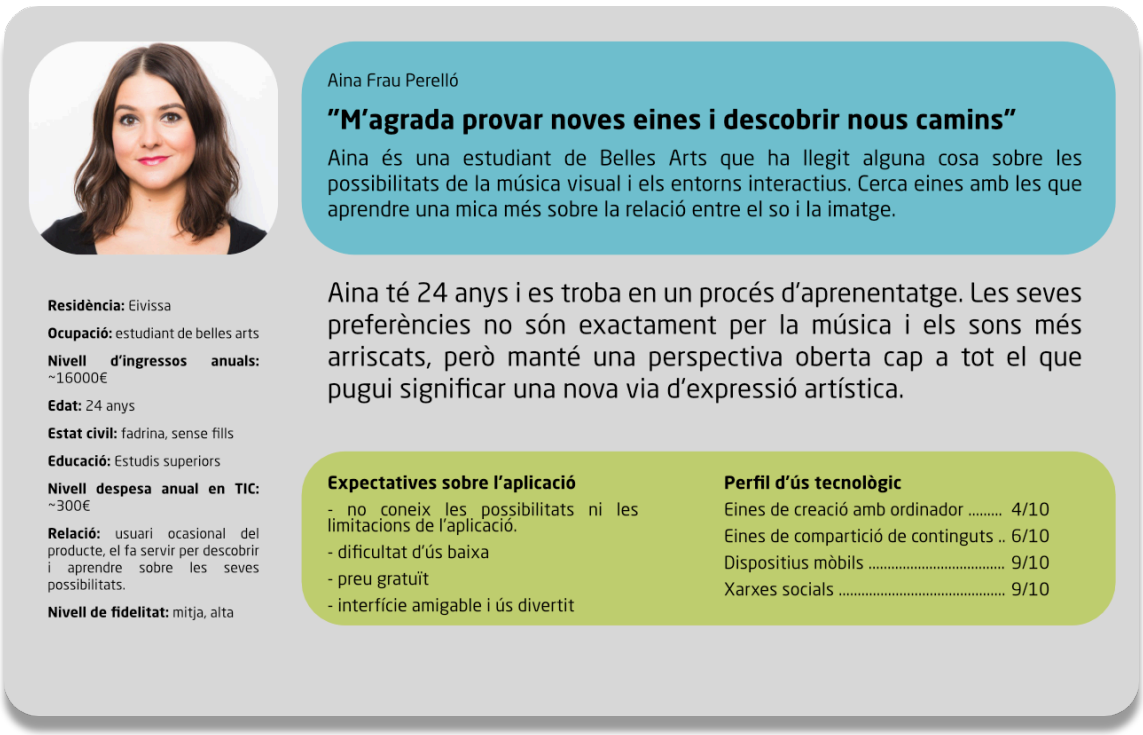
14.2 Persones

14.2.1 Persona focal



Il·lustració 26 - Persona focal

14.2.2 Persona secundària



Il·lustració 27 - Persona secundària

14.3 Escenaris

14.2.1 Escenari corresponent a la persona focal

És un dissabte al vespre i Josep fa un concert al Museu d'art contemporani "Es Baluard" dins el cicle de músiques visuals. El públic no és molt nombrós, però pot reconèixer als habituals en aquest tipus de convocatòries. En Josep prepara el seu portàtil, engega l'aplicació i connecta la sortida d'àudio de l'ordinador cap a la taula de mescles i, d'aquesta, cap el sintetitzador modular que processarà i enviarà el senyal de control cap al projector làser, ja en format ILDA.

Amb Quadramat, en Josep crea interessants patrons sonors i visuals, i aprofita els canvis de tema per carregar nous presets que havia programat la setmana anterior.

Quan arriba a casa, comparteix imatges i vídeos del concert, amb un enllaç a la web on es troba l'aplicació.

14.2.1 Escenari corresponent a la persona secundària

N'Aina ha descobert en un grup de Facebook una nova aplicació per crear sons i imatges que és gratuïta i que sembla poder fer una gran varietat de patrons molt interessants.

Després de llegar les instruccions, carrega alguns presets fets per altres usuaris i es passa una bona estona veient com funcionen i com canvien els sons i les imatges quan es modifiquen els paràmetres del sintetitzador.

Per entendre millor com funciona l'aplicació, cerca a la xarxa informació sobre les corbes de Lissajous, els polinomis Chebyshev i altres com la modulació en freqüència lineal. Semblen conceptes molt avançats, però amb l'avantatge que suposa poder visualitzar els canvis en els paràmetres del sintetitzador, i no només escoltar-los, ajuda a comprendre millor la seva relació amb els resultats obtinguts.

Quan arriba a l'escola, recomana als seus companys de classe l'aplicació i els indica que hi podran trobar alguns dels presets que ella ha programat.

15. Principis d'usabilitat

15.1 Consistència

Es facilita l'ús d'una aplicació quan es respecten els conceptes de disseny que s'han convertit en estàndards. La consistència implica la fixació d'unes constants representatives al llarg de tota una aplicació, de manera que el mateix tipus d'informació arribi a l'usuari sempre d'idèntica manera.

15.2 Prevenció d'errors

La GUI ha estat dissenyada de manera que només s'ofereixen a l'usuari les opcions que tenen cabuda dins el sistema (rangs de paràmetres vàlids, opcions dels menús desplegables per seleccionar opcions, etc.

Al procés de registre s'efectua una comprovació prèvia de l'e-mail d'alta de l'usuari i també una confirmació posterior.

15.3 Llei de Fitts

En aquest sentit, les opcions més importants tenen sempre més grandària o ser més visibles que les secundàries. Per exemple, els botons dels oscil·ladors són més grans que els de la matriu de modulació, per denotar una jerarquia en el procés de disseny sonor.

15.4 Metàfores

Es fan servir metàfores de les interfícies físiques d'instruments electrònics, com són els botons giratoris *("knobs")* i els interruptors *("toggles")*. La resta d'elements es corresponen amb la narrativa convencional dels entorns web: formularis, menús desplegables, etc.

15.5 Missatges d'error

Es preveu avisar a l'usuari d'una possible sortida involuntària de la pàgina actual, especialment probable en el cas dels dispositius mòbils. En aquest cas, el missatge és clar i no comporta cap connotació negativa o estressant.

Il·lustració 28 - Missatge d'error de l'aplicació

15.6 Usabilitat i disseny d'experiència d'usuari

15.6.1 Finestres o gràfics no sol·licitats

No es mostren en cap moment.

15.6.2 Augmentar la credibilitat del lloc

S'implementarà una pàgina de documentació amb FAQ, ressenya sobre l'autor i enllaços externs rellevants.

El disseny de l'aplicació resultar clar, modern i elegant.

15.6.3 Agrupació elements funcionals relacionats

D'una banda, es separa la secció corresponent als oscil·ladors de la de la matriu de modulació. També, cadascun dels oscil·ladors es troba separat dels altres, dins el seu propi requadre. Finalment, també els elements de cada oscil·lador es troben separats en sub-seccions com, per exemple, el selector de freqüència o el waveshaper.

També, la matriu de modulació presenta una estructura funcional adequada, amb totes les sortides de senyal com files i les entrades de senyal com columnes. El flux de senyal s'indica amb icones.

15.6.4 No carregar la memòria

L'aplicació no disposa de menús addicionals ni pàgines ocultes o orfes.

15.6.5 Descàrrega ràpida

La pàgina es descarrega amb molta rapidesa, ja que no ha de carregar ni imatges ni elements multimèdia amb un pes elevat.

15.7 Usabilitat relativa a maquinari i programari

S'ha comprovat l'aplicació en diversos sistemes operatius (Windows, OSX, Ubuntu, iOS i Android), així com en diversos tipus de dispositius (escriptori, tablet i mòbil) i també navegadors (Chrome, Firefox, Safari i Edge)

15.8 Usabilitat aplicada al disseny de navegació

Es fan servir senzills menús desplegables per seleccionar elements dels oscil·ladors.

Pel que fa al mapa del lloc, aquest resulta extremadament senzill, amb només tres pàgines (instrument, *login*, instruccions) no calen menús de navegació complexos ni estructures de navegació addicional com *breadcrumbs* o similars.

15.9 Usabilitat aplicada al disseny de les pàgines

S'ha tingut en compte els següents aspectes:

- interfície poc densa
- elements generadors a la part superior (oscs)
- densitat visual baixa
- elements ben alineats

- longitud de pàgina adequada
- logotip o nom instrument (implementat en el prototip #2)

15.9 Ús de la jerarquia visual per optimitzar la usabilitat

S'han tingut en compte els següents criteris de jerarquia visual:

- grandària: diferents mides dels botons, dels cossos de lletra, etc.
- Posició: disposició jerarquitzada dels elements: oscil·ladors de dreta a esquerra i matriu de modulació en posició inferior. La reixeta *responsive* situa tots els elements un sota l'altre en dispositius amb pantalles de mides reduïdes.
- contrast/color: es fa servir per indicar clarament la posició de cada botó rotatori.
- agrupació: es fan servir capses pels mòduls (oscil·ladors, matriu...) i sub-capses amb seccions mòduls (selector de freqüència, *waveshaper*...).

15.10 Ús d'imatges

L'ús d'imatges a l'aplicació és molt limitat. Només es fan servir icones d'elements GUI a partir d'imatges en format SVG. Els seus contorns són definits i el seu contrast suficient.

15.11 Barres de desplaçament:

En el prototip #1 queda pendent d'implementar alternativa a la barra de desplaçament horitzontal per la matriu de modulació en pantalles de mòbils. S'estudiaran quines alternatives poden donar millor resultat.

15.12 Llegibilitat

S'ha assegurat que les mides de les fonts són adequades assignat al text més petit una mida de 14px.

No es fan servir contrastos de color entre els elements de l'aplicació que dificultin la lectura. A un fons clar sempre li correspon un text fosc i viceversa.

16. Seguretat

Els riscos per la seguretat seran analitzats i detallats en el moment d'afrontar el desenvolupament del prototip #3, que incorpora la secció de back-end.

A títol preliminar, s'ha d'assegurar que el servidor no serà susceptible d'atacs com els de finals de 2016, que es varen conèixer amb el nom de "MongoDB apocalypse" (59). Algunes maneres de prevenir aquest tipus d'atacs passen per fers servir les versions més avançades del programari del back-end, així com claus d'administrador segures (60).

En el moment actual, el codi executat en el navegador no comporta riscos de seguretat coneguts.

16.1 Checklist de la OWASP *(Open Web Application Security Project)*

Es recomana fer referència a projecte OWASP com un punt de partida per reduir els riscos de seguretat de l'aplicació en línia (61).

16.1.1 Injecció

Aquest atac implica l'explotador trencant un context de dades convertint-lo en un context de codi fent servir caràcters amb codis especials.

16.1.2 Scripting entre llocs (XSS)

Aquest atac és una forma d'injecció, amb el navegador utilitzat per enterrar o amagar l'atac.

16.1.3 Autenticació trencada i gestió de sessió

Aquest atac implica l'explotador robant o assumint la identitat de les credencials d'autenticació no protegides d'un usuari.

16.1.4 Referències d'objectes directes insegurs

Es pot produir un risc d'exposició quan hi ha una referència a un objecte en una URL (un objecte com ara un fitxer, un directori, un registre de base de dades o una clau) o un paràmetre de formulari ja que un explotador podria canviar aquestes referències d'objectes directes i intenteu accedir a un fitxer diferent, no autoritzat, registre o clau de base de dades.

16.1.5 Cross-Site Request Forgery (CSRF)

Es produeix quan s'engana al navegador de l'usuari iniciant sessió en un lloc amb les credencials d'una altra persona.

16.1.6 Configuració incorrecta de seguretat

Es poden explotar les configuracions incorrectes del servidor d'aplicacions, així com la base de dades i la plataforma de base de dades subjacents a l'aplicació, especialment quan

s'implementen amb la configuració per defecte, coneguda públicament. Els atacs poden ser extremadament variats a causa de les moltes configuracions que es podrien configurar erròniament..

16.1.7 Emmagatzematge criptogràfic insegur

Els explotadors poden prendre o canviar dades que no estan protegides, com per exemple informació de targetes de crèdit, nombres de la Seguretat Social o credencials d'autenticació si no s'han fet servir un xifratge o hashing forts.

16.1.8 Falla en restringir l'accés a l'URL

Les aplicacions que no tenen control de control d'accés cada vegada que s'accedeix a una pàgina poden permetre als atacants falsificar URLs per accedir a les pàgines que es creu que estan ocultes.

16.1.9 Protecció de la capa de transport insuficient

Quan les aplicacions no autentiquen, codifiquen o protegeixen la confidencialitat i la integritat del trànsit de la xarxa sensible a atacs mitjançant un ús adequat dels certificats, el trànsit de dades pot ser interceptat per una tercera part amb intencions malicioses.

16.1.10 Redireccions i reenviaments no vàlids

Permet als atacants redirigir l'usuari a un lloc al que no volen anar i, potser, demanar informació personal fent que l'usuari faci pensar que es troba en un lloc web de confiança vàlid i familiar.

17. Tests

17.1 Tests d'usabilitat

Entre els beneficis dels test d'usabilitat destaquem els següents (62)

- Esbrinar si els participants poden completar les tasques especificades amb èxit
- Identificar quant de temps triga a completar les tasques especificades
- Descobrir si els participants queden satisfets amb el lloc web o amb el producte
- Identificar els canvis necessaris per millorar el rendiment i la satisfacció de l'usuari
- Analitzar el rendiment per veure si compleix els vostres objectius d'usabilitat

A partir de la implementació del prototip #3, quan el motor de síntesi estigui finalitzat i també funcioni el sistema de presets, es conduiran els primers tests d'usuari (63).

Es tractarà de tests moderats i en els que s'enregistrarà en vídeo l'activitat de l'usuari per poder treure'n les conclusions oportunes.

Es realitzaran tests a usuaris tant de tipus primari com secundari.

El procediment del test consistirà en demanar a l'usuari que realitzi una sèrie de tasques com, per exemple, donar-se d'alta com usuari, experimentar les possibilitats de creació sonora i visual de l'aplicació, crear alguns presets d'usuari i, finalment, compartir-los amb terceres persones. En acabar aquestes tasques se li demanarà a l'usuari que empleni unes preguntes relatives al producte i la seva usabilitat.

La finalitat dels tests és conèixer els punts febles del desenvolupament per eliminar aquells punts que dificultin l'ús de l'aplicació en el seu màxim potencial.

17.1.1 Contingut del test d'usabilitat

Es decideix realitzar només una fase de proves d'usabilitat i aquesta té lloc en una fase ja avançada del projecte. Idealment, s'haurien fet també altres proves d'usabilitat en les primeres fases del project i, fins i tot, abans que de començar el procés de codificació. Aquestes proves "sense codi" es poden fer sobre una interfície simulada dissenyada amb programari específic o també mitjançant l'ús de plantilles retallades en paper.

El test final d'usabilitat es realitza situant als usuaris davant de la pantalla de diferents dispositius: mòbil, tauleta i ordinador portàtil. Se'ls entrega un full amb una seqüencia d'instruccions i s'enregistra com les porten a terme. Poden parlar amb el responsable de la prova, però les seves interaccions quedaran especialment enregistrades. En acabar la prova se'ls entrega un full amb una sèrie de preguntes sobre la seva experiència.

1. Accedeixi a la pàgina *tfg.graumultimedia.com*
2. Registri's d'alta com usuari. Haurà d'indicar la seva adreça de correu electrònic i una contrasenya de la seva elecció
3. Faci servir els botons *prev/next* de l'aplicació web per navegar pels presets existents
4. Carregui un preset de la llista
5. Provi a cercar tots els presets que continguin la paraula *"moon"* i seleccioni'n un.
6. Provi a crear la seva pròpia composició sono-visual i guardi-la amb un nom de preset de la seva elecció.
7. Canviï el nom del preset anterior
8. Faci algunes modificacions i guardi-les amb un altre nom
9. Esborri el primer preset que ha guardat.

Taula 5 - Text amb la seqüència d'instruccions d'usabilitat

1. Quin és el seu nivell de coneixements de síntesi sonora? Ha fet servir mai un sintetitzador musical? Ha fet servir abans una consola de gràfics làser?
2. Com puntuaria d'1 a 10 el procés de càrrega de la pàgina? ____ Anoti les seves observacions al respecte.
3. Com puntuaria d'1 a 10 el procés de d'alta com usuari? ____ Anoti les seves observacions al respecte.
4. Com puntuaria d'1 a 10 el procés de càrrega i navegació pels presets? ____ Anoti les seves observacions al respecte.
5. Com puntuaria d'1 a 10 el procés de creació dels seus patrons sonors i visuals? ____ Anoti les seves observacions al respecte.
6. Ha comprés com funciona i quina és l'estructura dels controls de la interfície d'usuari (els botons a la pantalla)?
7. Com puntuaria d'1 a 10 el procés d'enregistrament dels presets? ____ Anoti les seves observacions al respecte.
8. Com milloraria l'aspecte visual de l'aplicació?
9. Com milloraria el procés de creació de patrons sono-visuals?
10. Anoti qualsevol observació o suggeriment

Taula 6 - Text amb les preguntes als usuaris

17.1.2 Conclusions del test d'usabilitat

Després d'observar com els usuaris fan servir l'aplicació, es pot confirmar que el seu ús no presenta dificultats a nivell bàsic, més enllà de les habituals en qualsevol aplicació web.

Com és habitual, s'observa un moment de dubte en el procés de *Login/Register*, possiblement a causa de la terminologia en anglès. No obstant, es descarta de moment realitzar la localització de l'aplicació i es deixa per més endavant un possible procés de traducció.

Els processos de navegació pels presets i cerca resulten exitosos i no plantegen cap dubte sobre el seu funcionament. S'han fet servir les convencions habituals en aquest tipus d'interaccions (selectors, botons, etc.) i aquest fet es reflecteix positivament.

El procés d'alta com usuari també resulta sense incidències.

En usuaris sense cap tipus d'experiència es pot observar que en pocs moments ja realitzen les seves pròpies figures i patrons, encara que provant a cegues els paràmetres a modificar. Aquest procés de recerca aleatòria els resulta engrescador i productiu. Els usuaris amb més experiència indiquen que la matriu de modulació els resulta fàcil i entenedora.

Entre els suggeriments dels usuaris corresponents amb la persona secundària destaquem la possibilitat d'escollir el color de les figures creades pel visualitzador, mentre que entre els que es corresponen amb la persona primària fan més referència als aspectes de control i als efectes sonors.

En resum, es considera que la usabilitat de l'aplicació es correspon amb els objectius establerts i no es detecten greus problemes que facin poc pràctic el seu ús. Es pren nota dels suggeriments respecte de la disposició dels elements del formulari de registre i s'implementaran en la versió definitiva.

17.2 Tests de seguretat

17.2.1 Injeccions de codi

Després d'efectuar uns tests bàsics de seguretat es comprova que l'aplicació pot ser sotmesa a injeccions de codi i es modifica el codi mitjançant un procés conegut com "desinfecció d'HTML" (64) que consisteix en eliminar les etiquetes considerades insegures en el moment que són passades del formulari al servidor.

17.2.2 Contrasenyes de l'entorn de desenvolupament (GitHub)

La contrasenya d'accés a la base de dades no queda emmagatzemada conjuntament amb el codi del repositori sinó que es fa servir una variable local de sistema a Heroku o, en el seu defecte quan el programa s'executa localment, els valors emmagatzemats al fitxer `local.js`

17.2.3 Protecció de les contrasenyes dels usuaris

El procediment conegut com *hashing* (65) fa servir algorismes unidireccionals que converteixen una quantitat de dades arbitrària en una "empremta digital" de longitud fixa que no es pot invertir. També tenen la propietat que si l'entrada canvia fins i tot una mica, el *hash* resultant és completament diferent. Aquesta característica el fa molt indicat per protegir les contrasenyes, perquè permetrà verificar si la contrasenya de l'usuari és correcta però també la protegeix si el fitxer de les contrasenyes queda compromès.

Per protegir les contrasenyes dels usuaris s'ha fet servir la llibreria bcrypt (43).

```
bcrypt.hash(req.body.password, 10, function(err, hash) {
    // Store hash in database
    user.password = hash;
    // {...}
}
```

17.2.4 Protecció de la identitat dels usuaris

S'ha evitat enviar al client les dades senceres dels presets dels altres usuaris, ocultant el camp corresponent a l'email i mostrant només el nom d'usuari (o nickname), que no té perquè ser necessàriament un nom real. Per filtrar aquest contingut, al servidor s'ha fet servir el següent codi, que elimina condicionalment la propietat `p.autor.email` de l'objecte `presets`:

```
presets.map(function(p) {
    if(p.autor.email != req.session.email) {
        delete p.autor.email;
    }
    return p;
});
```

18. Requisits d'instal·lació/implantació/ús

18.1 Requisits de maquinari

18.1.1 Maquinari del client per executar l'aplicació

Qualsevol dispositiu capaç de connectar-se a Internet i fer córrer la versió més actual del sistema operatiu mes actual compleix amb els requeriments mínims de maquinari per executar l'aplicació. Aquesta llista inclou dispositius com telèfons mòbils i tauletes capaços d'executar els sistemes operatius iOS 11 o Android 8.0; així com ordinadors portàtils i d'escriptori capaços d'executar Windows 10 o OSX High Sierra. La següent pagina (66) ens proporciona una llista detallada dels requeriments. El maquinari ha de disposar d'interfície de so integrada o externa.

18.1.2 Maquinari addicional per projeccions làser

Encara que l'aplicació es pot fer servir només amb qualsevol dispositiu dotat de pantalla i interfície de so, les seves possibilitats es fan evidents quan es connecta a un projector làser. Per realitzar la següent connexió són necessaris els següents elements:

- Tarja de so amb sortida "DC-coupled". Això permetrà voltatges constants en les sortides d'àudio, el que implica que el senyal es pugui quedar estacionari en un punt arbitrari. Algunes targetes bloquegen el senyal de corrent continu per protegir els altaveus, encara que es poden trobar moltes que no ho fan, principalment entre les destinades al mercat professional. Una llista d'aquestes targetes es pot trobar a: http://www.expert-sleepers.co.uk/siwacompatibility.html
- Interfície ILDA que permeti connectar la sortida de la targeta d'àudio al projector làser i generar els voltatges d'offset corresponents als colors RGB. Un exemple d'aquesta interfície és el mòdul Cyclops (67) de l'empresa LXZ Industries.

18.1.3 Maquinari del servidor

L'aplicació s'executa en un servidor virtualitzat a Heroku, per tant els requeriments de programari per part del servidor no s'apliquen en aquest cas.

18.2 Requisits d'instal·lació

18.2.1 Instal·lació del client

No calen instruccions específiques. Només cal visitar la URL de l'aplicació.

18.2.2 Instal·lació en el servidor

Les instruccions d'instal·lació en el servidor (Deployment) han estat detallades més a dalt, en l'apartat corresponent del procés de treball.

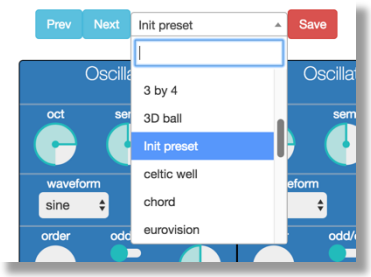
19. Instruccions d'ús

Quadramat. Guia d'usuari de l'aplicació.

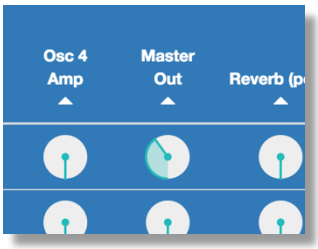
1. Amb el navegador web, visiteu la pàgina https://tfg.graumultimedia.com

2. Seleccioneu els paràmetres oportuns per crear els patrons sono-visual desitjats. A continuació s'indica pas a pas el procediment per crear una figura de complexitat mitja amb dos oscil·ladors:

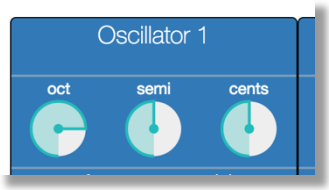
a. En primer lloc cercarem i carregarem el preset "Init Preset". Aquesta acció es pot dur a terme clicant sobre el selector a la part superior de l'aplicació on hi posa "Select a preset..." i a continuació escrivint "init" fins que només surti el nom del preset desitjat a la llista, després el seleccionarem fent novament clic sobre el nom complet del preset "Init Preset". També es pot fer un desplaçament dels elements de la llista i seleccionar-lo directament.



b. En haver carregat el preset seleccionat en el punt anterior sentirem un so greu i en el visualitzador es podrà veure un cercle. Podem pujar o baixar el seu volum pujant el valor del dial de la matriu de modulació de la primera fila, corresponent a Osc1 i a la penúltima columna, corresponent a la sortida "Master Out".



c. Amb els dials rotatoris "oct", "semi" i "cents" seleccionarem la freqüència de l'oscil·lador 1. Observarem com es pot canviar l'altura del to que s'escolta.

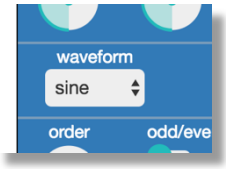


d. Seleccionarem un valor molt baix del dial “oct”. Observarem com el desplaçament és tan lent que es pot seguir el moviment del punt de llum a la pantalla i que ara el so té una freqüència tan baixa que no es pot sentir.

e. Tornem a seleccionar un valor més elevat pel dial “oct” fins que es torni a sentir un so.

f. Canviem la forma d’ona de l’oscil·lador amb el menú desplegable “Waveform”. Observarem que a cada forma d’ona li correspon un timbre sonor i una representació visual diferents. Finalment escollirem “square3”, obtenint una ona quadrada amb només els seus primes tres harmònics.

g. Canviem el valor del dial “order”. Observarem com canvia el timbre sonor i també la seva corresponent representació visual. Escollirem una combinació de valor del dial “ordre” i del interruptor “odd/even” que ens resulti interessant.



h. A la matriu de modulació, canviarem els valors dels dos primers dials, començant per la dreta i per la part superior, i observarem com el primer dial canvia la quantitat de modulació de l’oscil·lador 1 cap a l’oscil·lador 1, és a dir, cap a ell mateix; mentre que el segon dial fa el mateix amb la quantitat de modulació en amplitud. De moment, deixarem els dos valors al mínim.

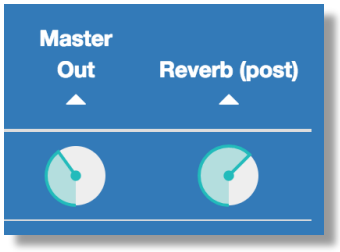
i. A l’oscil·lador 2 canviarem el valor del paràmetre “oct” per deixar-lo al mínim.

j. A la matriu de modulació, posarem a la posició de les 3 el dial corresponent a la modulació d’amplitud de l’oscil·lador 2 cap a l’oscil·lador 1; és a dir, el corresponent a la segona fila i la segona columna. Observarem com la figura canvia sola el seu patró i les seves mides mentre que el so presenta una lleugera variació tímbrica de tipus cíclic.

k. Podem experimentar lliurement amb la resta de paràmetres dels altres oscil·ladors i els valors de la matriu de modulació. Observarem com es poden crear una gran varietat de sons i figures.

l. Podem baixar el volum general de l’oscil·lador 1 amb el penúltim dial de la primera fila, a la columna etiquetada amb el nom “Master Out”. Observarem com el volum afecta a les mides de la figura en el visualitzador.

m. Finalment, amb el dial de la columna “Reverb (post)” afegirem a l’oscil·lador 1 la quantitat d’efecte de reverberació desitjada. Observarem que aquest efecte simula la presència de la font sonora dins un espai gran, com una sala de concerts o una església. És aconsellable anar variant els valors dels altres dials per escoltar millor aquest efecte.



3. Per navegar pels presets ja existents només cal prémer els botons Prev i Next a la part superior de la pantalla. També es pot seleccionar un preset directament a la llista, com s’ha fet en el punt anterior, i també es pot cercar un preset pel seu nom per seleccionar-lo després.

4. Si es vol guardar una determinada configuració dels botons a la pantalla, per recuperar-la posteriorment, es pot fer mitjançant el botó “Save”. Però abans de poder efectuar aquesta acció cal donar-se d’alta com usuari a l’aplicació. Després de fer clic sobre el botó “Register” l’aplicació ens portarà a la pàgina d’alta d’usuaris on introduirem el nostre email, nom d’usuari i contrasenya. Si el procés es porta a terme correctament se’ns mostrarà la pagina de benvinguda.

5. Ara ja podrem guardar qualsevol configuració de botons en forma de preset fent clic sobre el botó “Save As...” i donant un nom al preset. Veurem com aquest preset és ara accessible a la llista i per qualsevol persona que faci servir l’aplicació.

6. Amb els presets que hàgim fet nosaltres hi podrem fer accions com renombrar-los (botó “Rename”), modificar-los i guardar-los amb el mateix nom (botó “Save”), modificar-los i guardar-los amb un altre nom (botó “Save as...”) i esborrar-los definitivament (botó “Delete”).

7. Amb els presets fets pels altres usuaris només podrem fer servir l’opció “Save as...” i amb un nou nom figuraran com presets nostres.

8. Finalment, si ho desitgeu podeu sortir de la sessió. En cas de no fer-ho, la propera vegada que us connecteu a l’aplicació hi trobareu la sessió encara oberta i no caldrà que torneu a fer log-in. La sessió caducarà automàticament passades quatre setmanes.

20. Bugs

20.1 Bugs presents a la finalització del prototip #1

Els errors de programari més rellevants o decisius per la correcta execució de l'aplicació s'han anant solucionant en el procés de desenvolupament.

Així i tot, queden per resoldre els següents bugs en la data de 31 d'Octubre de 2017

1. ~~**En canviar l'ordre del *waveshaper* es produeixen petits clics.** Gravetat: baixa. Prioritat: baixa. Previsió de correcció: abans de finalitzar el prototip #2~~
2. ~~**En canviar el paràmetre "x/y/xy" es perd la coherència de fase entre els oscil·ladors.** Gravetat: alta. Prioritat: mitja. Previsió de correcció: abans de finalitzar el prototip #2~~
3. **El canvas no es redimensiona quan canvien les mides de la pantalla (possible canvi de posició en pantalles desktop?).** Gravetat: baixa. Prioritat: baixa. Previsió de correcció: abans de finalitzar el prototip #2

S'espera que els tests d'usabilitat realitzats en haver finalitzat els prototips #2 i #3 ajudin a trobar errors de programari encara no detectats.

20.2 Bugs presents a la finalització del prototip #2

S'han solucionat els errors 1 i 2 de l'apartat anterior. En canvi, encara queda sense resoldre el corresponent al redimensionament del canvas que, en la darrera versió, presenta noves disfuncionalitats.

4. **En els dispositius mòbils de pantalla més petita les dimensions del canvas de la visualització no són correctes**, presentant un excés d'altura que impedeix veure les figures correctament. Gravetat: mitja. Prioritat: baixa. Previsió de correcció: abans de finalitzar el prototip 3.

20.3 Bugs coneguts en el prototip final

No queden bugs coneguts pendents de solucionar. En el procés de proves amb usuaris no s'han detectat errors en el programari.

21. Projecció a futur

En la procés de realització del present treball s'han anat descobrint possibles millores i usos alternatius del programa que, a causa de les limitacions pròpies del projecte, no s'han pogut afrontar en primera instància.

21.1 Motor de síntesi

En primer lloc, senyalarem les funcionalitats addicionals que seria recomanable implementar de cara al motor de síntesi i la seva interfície gràfica d'usuari, que donarien noves possibilitats performatives a l'instrument, especialment de cara a la interpretació en directe.

1. Control de la quantitat global de modulació cap a cadascun dels paràmetres de destinació mitjançant l'ús d'atenuadors (semblants en funcionalitat a les rodes de modulació dels sintetitzadors convencionals) (68). Requeriria canvis profunds en l'estructura del motor de síntesi i nous enrutaments, a més de afegir nous elements a la interfície d'usuari.
2. *Morphing* entre dos presets mitjançant un controlador de cinta virtual (69) que permetria una interpolació lineal entre els valors de dos presets arbitraris. Permetria efectuar canvis dràstics de sons/figures de forma senzilla i facilitaria el procés de transició entre presets en un entorn performatiu. Requeriria la implementació d'un doble selector de presets, canvis en el motor de síntesi per fer que tots els paràmetres fossin numèrics (eliminant les llistes) i afegir l'element corresponent a la GUI.
3. Control via MIDI de tots els paràmetres. Aquesta és la funcionalitat més senzilla d'implementar ja que només caldria incloure una pàgina de configuració i un enllaç entre els valors llegits via MIDI i la interfície gràfica. Permetria modificar els paràmetres de forma més còmoda i obtenir una realimentació no només visual sinó també tàctil de la posició de cada paràmetre. Es recomana fer servir controladors MIDI amb USB amb botons de tipus "*rotary encoder*", ja que en tot moment permeten manipular els paràmetres del so a partir del seu valor actual, sense canvis abruptes.

21.2 Sistema de gestió de presets

En segon lloc, es detallen aquelles funcionalitats que seria interessant incloure en el sistema de gestió de presets i que facilitarien la navegació entre els preajustaments dels usuaris, la seva classificació i publicació.

1. Botó de compartir preset que retorna una URI a la que es pot accedir directament. Possibilitat d'integració d'aquesta característica amb les xarxes socials més conegudes. Seria relativament fàcil d'implementar ja que només implicaria la programació d'una finestra modal amb les opcions de les xarxes en les que compartir i una mínima programació amb una ruta addicional de part del servidor que rebés l'identificador del preset i retornés la pàgina principal amb el preset carregat.

2. Sistema de categories que permetés a l'autor de cada preset assignar-hi una paraula clau que defineixi les seves principals característiques: estrident, greu, electrònic, seqüenciat, progressiu, etc.
3. Sistema de votacions per avaluar la popularitat dels presets i assignar puntuacions amb un sistema d'una a cinc estrelles o amb un sistema de *"likes"*. Per evitar que el mateix usuari votés més d'una vegada el mateix preset caldria implementar una col·lecció amb les votacions realitzades per cada usuari. Aquest sistema permetria mostrar automàticament la puntuació donada per l'usuari actual a cada preset en carregar-lo.
4. Sistema de cerca de presets filtrat per categories i popularitat. Substituint el selector actual per una finestra modal amb funcionalitat completa es podria implementar un sistema de gestió de presets més avançat que permetés aquestes i altres noves maneres d'accedir i modificar la llista de presets.

21.3 Sistema de gestió d'usuaris

Finalment, de cara al sistema de gestió d'usuaris, quedaria per implementar funcionalitat addicional que es podria considerar bàsica i que afecta sobre tot a qüestions de seguretat.

1. *Captchas* per intentar controlar el registre massiu d'usuaris i l'ús maliciós per terceres parts.
2. Confirmació de l'alta dels usuaris mitjançant correu electrònic, per assegurar que aquest s'ha introduït correctament i que l'usuari té accés al compte.
3. Recuperació i canvi de contrasenya per part dels usuaris fent servir la funcionalitat bàsica del punt anterior.
4. Pàgines de perfil d'usuari, tant públiques com privades, amb els seus vots emesos i rebuts, dades personals, etc.

22. Pressupost

Encara que la naturalesa del projecte no ha fet necessari el pagament d'hores de treball, ni tampoc lloguer o compra d'equipament, es procedeix a la seva valoració amb la finalitat de poder avaluar els aspectes econòmics del desenvolupament del producte.

22.1 Equip Humà

Es considera que la dificultat del projecte es pot assolir amb un equip humà format per un sol programador web full-stack. Es prenen com referències del salari les indicacions de les pàgines web (70) i (71) que situen el cost laboral d'un desenvolupador *full-stack* entre els 26.000€ i els 37.000€, havent-hi ofertes que arriben fins als 50.000€ anuals.

El càlcul del preu brut per hora treballada per hora s'ha efectuant tenint en compte 46 setmanes de treball efectiu per any i jornades de 40 hores setmanals.

*46 setmanes * 40 hores = 1.840 hores/any*

36.000€ / 1840 hores = 19.56€ / hora treballada

Cost Seguretat Social i altres despeses: +30%

Cost final: 25,43€ / hora treballada

El temps de desenvolupament del projecte és de 13 setmanes a mitja jornada

*13 setmanes * 20 hores = 260 hores*

El cost total de les hores de treball és, per tant, el següent:

260 hores * 25,43€/hora = 6.611,8€

22.2 Equipament tècnic

Es considera un equipament tècnic format pels següents elements:

1. ordinador d'escriptori: 2.000€
2. tauleta: 600€
3. telèfons mòbils iOS i Android: 1000€
4. taula de mescles: 300€
5. altaveus: 2000€
6. sintetitzador modular: 2400€
7. projector làser: 1000€

El material detallat puja a un import de 9.300€.

No existeixen referències per poder calcular l'import del lloguer, però procedint a una aproximació que tingui en compte la vida mitja útil dels productes, el seu valor al final de la vida útil i el cost de manteniment, es procedeix a la següent estimació:

Vida útil: 5 anys, és a dir 60 mesos.

Valor residual dels productes: 10%

*Valor a amortitzar: 9300 * 90% = 8.370€*

Quota mensual d'amortització: 8.370€ / 60 = 139,50€/mes

Benefici industrial i costos de l'operació de lloguer: +30%

Quota mensual d'amortització final: 139,50 + 30% = 181,35€/mes

El cost total imputable al lloguer de l'equipament tècnic és, per tant, el següent:

181,35€/mes * 3 mesos = 544,05€

22.3 Espai de treball

El lloguer d'un espai de *coworking* a Palma (Mallorca) amb les següents característiques és de 150€/mes + IVA:

- lloc fix de treball a jornada completa
- L-V 09:00 – 19:00
- 10h de sales de reunions
- Taquilla

150,00€/mes * 3 mesos = 450€

22.4 Lloguer i serveis de servidor

Heroku ofereix un servei de facturació basat en l'ús dels *dynos* com una abstracció d'un model de contenidor d'aplicacions. Els *dynos* són contenidors Linux virtualitzats i aïllats dissenyats per executar codi seguint els comandaments de l'usuari. Les aplicacions poden escalar fins qualsevol nombre de *dynos* segons les seves necessitats de recursos.

Les tarifes de Heroku ens ofereixen la possibilitat de escollir entre models de facturació que van des del model gratuït amb un nombre limitat d'hores d'ús mensual fins a tarifes d'alt rendiment de 500€/dyno/mes.

D'entrada, es proposa fer servir el model gratuït i avaluar un canvi de model de pagament segons evolucionin les necessitats.

Gratuït, amb restriccions

Concepte	Preu/unitat	Unitats	Preu
Programador full-stack	25,43€/hora	260 hores	6.611,80€
Lloguer equipament tècnic	181,35€/mes	3 mesos	544,05€
Espai de treball	150,00#/mes	3 mesos	450,00€
Lloguer i serveis servidor	Gratuït	-	-
Total			**7.605,85€**

Taula 7 . Resum del pressupost

23. Llicència del programari

23.1 Definició i tipus llicències lliures.

El programari lliure és aquell que dóna a l'usuari la llibertat de compartir-lo, estudiar-lo i modificar-lo. Rep el nom de programari lliure perquè allibera l'usuari.

23.1.1 Programari lliure

La cita anterior, extreta d'una publicació de la *Free Software Foundation* (72), ens resumeix els punts fonamentals que defineixen el que és el programari lliure. La definició de programari lliure (73) va ser escrita per Richard Stallman i enumera quatre llibertats:

0. La llibertat d'executar el programa com es desitgi, amb qualsevol propòsit.
1. La llibertat d'estudiar com funciona el programa i modificar-lo per què funcioni segons els propis desitjos. Un accés lliure al codi font és una precondició d'aquesta llibertat.
2. La llibertat de redistribuir còpies del programa per ajudar als altres
3. La llibertat de distribuir còpies de les versions pròpiament modificades. D'aquesta manera se n'assegura que la comunitat es pugui beneficiar dels canvis. L'accés lliure al codi font és també una precondició d'aquesta llibertat.

A continuació es detalla una llista de les llicències més habituals que es poden considerar de codi lliure (74):

- Llicència Pública General GNU (GPL GNU) (recomanada)
- Llicència Pública Menys General (LGPL GNU)
- Llicència de Guile
- Llicència de les unitats d'execució del compilador GNU Ada
- Llicència X11
- Llicència original del BSD
- Llicència BSD modificada
- Llicència d'Apache
- La llicència de Zope
- Llicència pública IBM
- Llicència pública del projecte LaTeX
- Llicència de Perl
- Llicència pública de Mozilla (MPL)

23.1.2 Programari de codi obert

L'etiqueta "codi obert" cerca desemmarcar-se de les motivacions ètiques i filosòfiques del programari lliure (75). Es va escollir l'etiqueta "codi obert" per representar el programari que es pot compartir lliurement en l'esperança que pogués reflectir millor el valor de negoci d'un model de desenvolupament col·laboratiu i dirigit per la comunitat.

Aquests són els deu punts que ha de complir un programari que es vulgui considerar de codi obert:

1. Redistribució gratuïta: la llicència no ha de restringir a cap de les parts de vendre o regalar el programari com a component d'una distribució de programari més gran que contingui programes de diverses fonts.
2. Codi font: el programa ha d'incloure el codi font i ha de permetre la distribució en codi font, així com el formulari compilat.
3. Obres derivades: la llicència ha de permetre modificacions i treballs derivats, i ha de permetre que es distribueixin en els mateixos termes que la llicència del programari original.
4. Integritat del codi font de l'autor: la llicència pot restringir el codi font que es distribueixi de forma modificada només si la llicència permet la distribució de "fitxers de pegats" *("patch files")* amb el codi font per tal de modificar el programa en temps de compilació.
5. No discriminació contra persones o grups: la llicència no pot discriminar contra cap persona o grup de persones.
6. No discriminació contra camps d'esforç: la llicència no pot restringir a ningú que faci ús del programa en un camp específic d'esforç *("field of endeavour")*.
7. Distribució de la llicència: els drets inherents al programa han d'aplicar-se a tots els que es redistribueix el programa sense necessitat de que aquestes parts hagin d'executar una llicència addicional.
8. La llicència no ha de ser específica d'un producte: els drets adjunts al programa no han de dependre del fet que el programa sigui part d'una determinada distribució de programari.
9. La llicència no ha de restringir un altre programari: la llicència no ha de posar restriccions a un altre programari que es distribueix juntament amb el programari amb llicència.
10. La llicència ha de ser tecnològicament neutral: cap disposició de la llicència es pot basar en cap tecnologia individual o estil d'interfície.

Per tant, queda clar que els termes són en molts casos, encara que no sempre, intercanviables encara que els aspectes de la filosofia del codi lliure siguin més idealistes i els de la filosofia de codi obert més pragmàtics.

El fet que un programari pugui ser considerat com lliure o de codi obert dependrà de la seva llicència de distribució i de si l'esmentada llicència ha sigut aprovada per la Free Software Foundation, per la Open Source Initiative, o per les dues.

La llista de llicències aprovades per l'OSI inclou les següents (76):

- Apache License 2.0
- BSD 3-Clause "New" or "Revised" license
- BSD 2-Clause "Simplified" or "FreeBSD" license
- GNU General Public License (GPL)
- GNU Library or "Lesser" General Public License (LGPL)
- MIT license (escollida pel projecte)
- Mozilla Public License 2.0
- Common Development and Distribution License
- Eclipse Public License

23.2 Tipus de llicència escollida. Justificació.

Pel projecte actual s'ha escollit la llicència MIT (77). La llicència MIT és curta i permissiva amb condicions que només requereixen preservar el copyright i avisos de llicència. Els treballs llicenciats, modificacions i treballs majors poden ser distribuïts baix diferents termes i sense el codi font.

23.2.1 Permisos concedits:

- Ús comercial ✓
- Modificació ✓
- Distribució ✓
- Ús privat ✓

23.2.2 Limitacions

- Responsabilitat civil ✗
- Garantia ✗

23.2.3 Condicions

- Avís de llicència i copyright

23.2.4 Text de la llicència

MIT License

Copyright (c) 2017 Pere Amengual-Gomila

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN

ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

23.2.5 Justificació del tipus de llicència escollit

S'ha escollit la llicència MIT ja que és pràcticament una de les més senzilles i menys restrictives. La meva finalitat amb el projecte és de tipus educatiu i, en segon, terme evangelitzador en el sentit de donar a conèixer les possibilitats de creació sonora que ofereixen els navegadors web mitjançant la API Web Audio.

24. Conclusions

En primer lloc, la realització d'aquest Treball Final de Grau m'ha ajudat a recordar i replantejar-me el motiu pel que vaig iniciar els estudis del Grau en Multimèdia. Lluny de motivacions laborals o "inèrcia" del cicle educatiu, la meva principal motivació va ser aprendre a integrar les possibilitats de les Tecnologies de la Informació i la Comunicació amb els processos creatius que vaig iniciar fa ja molts anys en el camp de l'experimentació sonora.

Més enllà de la utilització d'eines que permeten l'edició de continguts audio-visuals que podríem qualificar com "convencionals" (vídeo, animació 3D, composició digital, etc.) el que m'interessa és la possibilitat de programar directament els dispositius per aconseguir la creació simultània de creacions sonores i visuals. La meva experiència prèvia amb sintetitzadors de so m'ha ajudat a avançar més ràpidament en la part del present projecte relativa a la programació d'un motor d'àudio (*audio engine*) que considero relativament sofisticat i lluny de les típiques aplicacions musicals més convencionals. A més, els coneixements adquirits en distintes assignatures del grau m'han ajudat a poder afrontar el repte de la creació simultània de sons i imatges, posant en pràctica coneixements en principi més abstractes, com les relacions trigonomètriques, per obrir el camí i investigar en disciplines que considero fascinants, com la música visual.

La fase inicial del procés de treball va consistir en una triple investigació sobre les possibilitats de la síntesi sono-visual, les característiques de les interfícies gràfiques dels instruments musicals virtuals i, sobre tot, les dificultats del desenvolupament d'un sistema de back-end per gestionar els presets i els usuaris de l'aplicació. D'aquestes tres, la darrera era la que menys coneixia i a la que he hagut de dedicar més temps, tot i que vaig preveure la dificultat d'aquesta fase d'investigació i vaig començar a preparar-la molt abans de l'inici del semestre actual. Podríem dir que, malgrat la seva dificultat tècnica, he afrontat la part del motor d'àudio i la GUI amb més alegria i menys esforç que la part relativa al back-end de l'aplicació, en la que tenia molta menys experiència.

El desenvolupament de l'aplicació en sí mateixa no ha patit més dificultats de les previstes i, la solució dels problemes que han anat sorgint sempre s'ha pogut resoldre afrontant de forma metòdica cadascun dels passos necessaris. El més problemàtic ha estat fer front a dificultats molt específiques de la plataforma en la que s'executa l'aplicació. Cal dir que, afortunadament, avui en dia es pot trobar moltíssima informació a la xarxa sobre qualsevol possible error o entrebanc que un es pugui trobar durant el procés de desenvolupament. Sona a acudit, però és ben cert allò de que "Google és el teu amic".

Valoro molt positivament els coneixements adquirits en el procés de realització del Treball Final de Grau i especialment el factor extra de motivació que suposa haver de desenvolupar un producte "complet" i llest per la seva utilització, més enllà d'un simple exercici o pràctica unidisciplinar per la qual només es disposa d'algunes setmanes.

Així i tot, m'hagués agradat disposar encara de més temps per poder afinar aquells aspectes que sé que haurien pogut millorar i que acabarien per fer del TFG un producte

acabat i definitiu. Encara que estic completament satisfet amb la feina realitzada, queda un bon camí per recórrer una vegada hagi posat l'aplicació a disposició del públic. M'agradaria millorar especialment aquells aspectes relatius a la publicació i promoció dels continguts creats amb l'aplicació. Per exemple, la possibilitat de que els usuaris puguin compartir a les xarxes socials captures d'imatge, vídeo o fins i tot un enllaç a l'aplicació que carregui directament un preset. En definitiva, el que diferencia una aplicació web d'una altra que es pugui realitzar sobre dispositius físics és la seva capacitat infinita per ser compartida i despertat tot tipus de processos de caire col·laboratiu. Sobre tot, espero que aquesta aplicació tingui un recorregut més enllà de l'àmbit acadèmic en el que ha estat concebuda.

En conclusió, considero que el repte de realitzar un projecte d'aquestes característiques és un element molt positiu i fins i tot fonamental en el llarg procés d'aprenentatge que es un grau com el que ara finalitzo. Ha suposat una motivació per anar més enllà de la zona de confort i, a més, un punt de partida cap a nous horitzons amb el recolzament que suposa el reconeixement de l'experiència acadèmica.

Annex 1. Lliurables del projecte

- Carpeta DOCUMENTACIÓ
  - Memòria en formats doc i pdf
  - Autoinforme avaluació en format doc i pdf
- Carpeta Projecte
  - Subcarpeta amb el codi font
  - Subcarpeta amb bases de dades
  - Subcarpeta amb altres documents (diagrama Gantt, etc.)
  - Document “Readme.pdf” amb enllaç a l’aplicació en línia i instruccions d’ús
- Carpeta Presentacions
  - Document amb enllaç a la presentació en vídeo sobre el projecte (<15 mins.) dirigida al tribunal d’avaluació. Lliurada amb l’eina Present@
  - Presentació escrita-visual en format *Powerpoint* + render a vídeo 720p
- Document amb enllaç a la publicació en l’O2 del projecte, la memòria, i els arxius de codi font.

Notes:

En el projecte no s’han fet servir recursos audiovisuals de terceres parts.

El codi de llibreries JavaScript i fulls d’estil de terceres parts es troba dins les carpeta del projecte `/public/javascripts` i `/public/stylesheets`.

Annex 2. Codi font (extractes)

Aquestes són alguns extractes de les parts més rellevants del codi:

Annex 2.1 Fragments de codi de l'oscil·lador

En aquest fragment podem veure com es crea l'objecte corresponent a l'oscil·lador `Osc1a` i, en el mateix procés, s'assignen alguns dels seus paràmetres inicials i es declara l'enrutament dels mòduls que el segueixen. És a dir, de l'oscil·lador el senyal passarà pel *waveshaper* `cheby0sc1a`, d'allà al `mergeOsc1.left`, i d'allà al visualitzador `waveformL`. Totes aquestes accions tindran lloc en un moment determinat pel valor de la variable temps, amb la finalitat de fer-les coincidir amb el moment de creació de l'oscil·lador `Osc1b`.

```
qm.engine.nodes = {
    //************* nodes osc1 ***********
    mergeOsc1: new Tone.Merge(),
    limiterAMOsc1: new Tone.Limiter(-24),
    chebyOsc1a: new Tone.Chebyshev(1),
    chebyOsc1b: new Tone.Chebyshev(1),
    osc1a: new Tone.Oscillator({
        "type": "sine", // forma d'ona inicial
        "volume": "-6", // unitats dB en creació
    "frequency": 100 // unitats Hz
    }),
    osc1b: new Tone.Oscillator({
        "type": "sine", // forma d'ona inicial
        "volume": "-6", // unitats dB en creació
    "frequency": 100 // unitats Hz
    })
}
```

Així es determinen les entrades de modulació dels valors de freqüència i amplitud mitjançant l'ús de bussos anomenats `osc1FreqChan` i `osc1AmpChan`, respectivament. En el cas de la modulació d'amplitud es fa servir un limitador per assegurar que els valors romandran en un rang estable; aquest limitador és on els altres oscil·ladors envien els seus senyals de modulació d'amplitud, que després són enrutats cap al paràmetre de volum dels dos components de l'oscil·lador de quadratura.

```
qm.engine.busses = function () {
    //************busses to osc1 *********
    qm.engine.nodes.osc1a.frequency.receive("osc1FreqChan");
    qm.engine.nodes.osc1b.frequency.receive("osc1FreqChan");
    qm.engine.nodes.limiterAMOsc1.receive("osc1AmpChan")
}
```

Annex 2.2 Fragments de codi de la GUI del selector de freqüència

En primer lloc, es defineix una escala justa amb les relacions de freqüències de les notes de l'escala respecte de la tònica (nota fonamental de l'escala). Per facilitar la integració amb la GUI, es defineix una escala que va des de la segona nota de l'octava anterior fins la darrera de l'octava actual dúna escala cromàtica.

```
Nexus.tune.createJIScale(
    8 / 15,
    9 / 16, // RE -1
    3 / 5,
    81 / 128, // MI -1
    2 / 3, // FA -1
    729 / 1024,
    3 / 4, // SOL -1
    4 / 5,
    27 / 32, // LA -1
    8 / 9,
    243 / 256, // SI -1
    1 / 1, // DO
    256 / 243,
    9 / 8, // RE
    32 / 27,
    5 / 4, // MI
    4 / 3, // FA
    1024 / 729,
    3 / 2, // SOL
    128 / 81,
    5 / 3, // LA
    16 / 9,
    15 / 8); //SI
```

A continuació, es defineixen els objectes de la GUI, amb els seus valors inicials, i els *listeners* que escoltaran els seus *events*.

```
qm.ui.e.dials = {
    //***************OSCIL·LADOR 1***************
  dialOsc1Oct: new Nexus.Dial('#osc1Oct', {
        'size': [50, 50],
        'interaction': 'vertical', // "radial", "vertical", or "horizontal"
        'mode': 'relative', // "absolute" or "relative"
        'min': -6, //
        'max': +6, //
        'step': 1,
        'value': 3 // valor inicial
    })
}


qm.ui.binds = function () {
  qm.ui.e.dials.dialOsc1Oct.on('change', qm.ui.changes.dialOsc1OctSemiCents);
  qm.ui.e.dials.dialOsc1Semi.on('change', qm.ui.changes.dialOsc1OctSemiCents);
  qm.ui.e.dials.dialOsc1Cents.on('change', qm.ui.changes.dialOsc1OctSemiCents);
}
v
```

Com podem veure, els tres *listeners* criden la funció `qm.ui.changes.dialOsc1OctSemiCents()` per calcular el valor de la freqüència que s'ha d'assignar a l'oscil·lador, ja que aquest és una combinació dels valors dels tres elements de la GUI. El codi de la funció que calcula el valor de la freqüència és el següent:

```
qm.ui.assignaValorFreq = function(dialOscOct, dialOscSemi, dialOscCents, osca,
oscb) {
    var frequenciaOscillador = Nexus.octave(dialOscOct.value - 4) *
Nexus.note(dialOscSemi.value) * Math.pow(Math.pow(2, 1 / 12),
dialOscCents.value);
    var temps = Tone.now() + 0.05; // per fer coincicir en el temps els canvis i
mantenir la coherència de fase
  osca.frequency.setValueAtTime(frequenciaOscillador, temps);
    oscb.frequency.setValueAtTime(frequenciaOscillador, temps);
}
```

El càlcul es realitza tenint en compte que la funció auxiliar `Nexus.octave()` retorna un multiplicador pels valors de les octaves. Per exemple `Nexus.octave(-1)` retorna 0.5 i `Nexus.octave(2)` retorna 4. Aquest valor es multiplica per la freqüència que ens proporciona `Nexus.note()`, que es correspon amb la nota de l'escala enviada com argument. El valor de la seva freqüència es veu alterat pel valor de `Nexus.tune.root` (que no hem modificat i té el valor per defecte) i també per les definicions de les relacions de freqüències especificades a la definició de l'escala justa que hem assignat inicialment. Finalment, per l'afinació de les centèsimes de semitò es multiplica pel valor de $(\sqrt[12]{2})^{cents}$, ja que el valor de cents té un rang [-1,1] i la relació entre els semitons és exponencial i n'hi ha 12 per octava. Recordem que la relació de freqüències de cada octava respecte de la anterior és 2:1.

Annex 2.2 Fragments de codi del visualitzador

El visualitzador fa servir l'element `<canvas>`. En primer lloc s'ha d'obtenir l'objecte del DOM i, a partir d'aquí, es poden definir algunes de les seves propietats i aplicar-hi mètodes per dibuixar-hi. Per aconseguir l'efecte de persistència dels oscil·loscopis analògics, a cada quadre la pantalla s'esborra dibuixant-hi un `fillRect()` amb un valor d'alfa baix, que després es restaura al valor inicial per dibuixar els altres elements. `valuesL[]` i `valuesR[]` són *arrays* que contenen els valors de la forma d'ona en un moment determinat. El que es fa es llegir els *arrays* i dibuixar línies que vagin d'un parell de punts (x,y) fins al següent fins acabar amb el *buffer* d'àudio. Després es tornen a dibuixar els valors actuals de la forma d'ona en el següent quadre d'imatge, i així, successivament.

```
qm.vis = {};

qm.vis.canvas = document.getElementById('visuals');
qm.vis.contenidorVisuals = document.getElementById("contenidorvisuals");
qm.vis.contenidorGUI = document.getElementById("GUI");
qm.vis.canvasCtx = qm.vis.canvas.getContext('2d');
qm.vis.aspectRatio = null; // Es calcula a posteriori, al init

qm.vis.dibuixa = function() {
    var valuesL = qm.engine.waveform.l.analyse();
```

```
    var valuesR = qm.engine.waveform.r.analyse();
    qm.vis.canvasCtx.globalAlpha = 0.15;
    qm.vis.canvasCtx.fillRect(0, 0, qm.vis.canvas.width, qm.vis.canvas.height);
    qm.vis.canvasCtx.globalAlpha = 1;
    qm.vis.canvasCtx.beginPath();
    qm.vis.canvasCtx.lineJoin = "round";
    qm.vis.canvasCtx.lineWidth = 1;
    for (var i = 1, len = valuesL.length; i < len; i++) {
        var valy = valuesL[i] / 255;
        var valx = valuesR[i] / 255;
        var x = valx * qm.vis.canvas.width;
        var y = valy * qm.vis.canvas.height * qm.vis.aspectRatio +
(qm.vis.canvas.height - qm.vis.canvas.width) / 2;
        qm.vis.canvasCtx.lineTo(x, y);
    }
    qm.vis.canvasCtx.stroke();
    requestAnimationFrame(qm.vis.dibuixa);
}

qm.vis.init = function() {
    qm.vis.contenidorVisuals.style.height =
String(qm.vis.contenidorGUI.offsetHeight) + "px";
    qm.vis.canvas.width = qm.vis.contenidorVisuals.offsetWidth;
    qm.vis.canvas.height = qm.vis.contenidorVisuals.offsetHeight;
    qm.vis.canvasCtx.globalAlpha = 1;
    qm.vis.canvasCtx.strokeStyle = "hsl(120, 100%, 50%)";
    qm.vis.canvasCtx.fillStyle = "black";
    qm.vis.canvasCtx.clearRect(0, 0, qm.vis.canvas.width, qm.vis.canvas.height);
    qm.vis.aspectRatio = qm.vis.canvas.width / qm.vis.canvas.height;
    qm.vis.dibuixa();
}
```

Annex 2.3 Exemples d'HTML dinàmic sota Pug (Jade)

`Layout.pug` funciona com a plantilla contenidora comuna a la resta de pàgines. A més de la sintaxi pròpia de Pug, podem observar que es dóna valor a unes variables javascript de forma dinàmica, depenent de si l'usuari ja ha iniciat la seva sessió en el servidor.

```
doctype html
html
  head
    meta(charset='utf-8')
    // per evitar que la pantalla faci scroll
    meta(name='viewport', content='width=device-width, initial-scale=1, maximum-
scale=1, user-scalable=no')
    title Quadramat - #{title}
    // fulls d'estils
    link(rel='stylesheet',
href='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css')
    link(href='https://cdnjs.cloudflare.com/ajax/libs/select2/4.0.6-
rc.0/css/select2.min.css', rel='stylesheet')
    link(rel='stylesheet', href='/stylesheets/estils.css')
    // llibreries js

script(src='https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js')

script(src='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js'
)
    script(src='https://cdnjs.cloudflare.com/ajax/libs/select2/4.0.6-
rc.0/js/select2.min.js')
    script(type='text/javascript', src='/javascripts/ui-master/dist/NexusUI.js')
    script(type='text/javascript', src='/javascripts/Tone.js-
master/build/Tone.js')
    script(type='text/javascript', src='/javascripts/StartAudioContext.js')
    script(type='text/javascript').
    - if(alreadyLoggedIn) {
      script(type='text/javascript').
        window.sessionStorage.email = "#{session.email}";
        window.sessionStorage.name = "#{session.name}";
    - }

  body
    block content
```

A `login.pug` podem observar com, depenent de si l'usuari ha iniciat o no la seva sessió, es presenta un contingut alternatiu. Més concretament, si ja ha iniciat la sessió el formulari només dóna opció a tancar la sessió i enviar la seva informació a la ruta `/login/logout` mitjançant POST; en canvi, si no ha iniciat encara la seva sessió, el formulari dóna opció a introduir email i contrasenya i enviar-los a `/login/check` també mitjançant POST. El servidor tractarà de forma convenient la informació rebuda, tancant o obrint la sessió, respectivament.

```
extends layout

block content
    - if(alreadyLoggedIn)
        form(action='/login/logout', method='POST')
            input(type='submit', value='Logout')
    - else
```

```
    - if(error)
        p.alert-danger L'usuari o el password són erronis.
    form(action='/login/check', method='POST')
        label E-mail:
        input(type='email', name='email', required='')
        label Password:
        input(type='password', name='password', required='')
        input(type='submit', value='Submit')
```

A `register.pug` podem trobar com, a més d'incloure el contingut de la plantilla layout, es comprova el valor de la variable `error`, que tindrà el valor `1` si s'ha arribat a la pàgina després d'una redirecció del servidor si aquest ha trobat que ja existeix un usuari amb l'email o el nom proporcionats. En altre cas, es proporciona a l'usuari el formulari amb el que podrà donar-se d'alta introduint el seu e-mail, nom i contrasenya. Queden pendents alguns punts menors de fàcil implementació com demanar a l'usuari que introdueixi dues vegades la contrasenya per assegurar que no ha comés cap error, i també d'altres d'una complexitat una mica major com oferir la possibilitat de canviar o recuperar la contrasenya si s'ha oblidat. Aquests punts es deixen pendents d'implementació a causa de les restriccions inherents a la data d'entrega del projecte i considerant que s'ha assolit ja un grau de funcionalitat suficient.

```
extends layout

block content
    - if(error)
        p.alert-danger Ja existeix un usuari amb aquest email o amb el mateix
nom!
    form(action='/register/new', method='POST')
        label E-mail:
        input(type='email', name='email', required='')
        label Name:
        input(type='text', name='name', required='')
        label Password:
        input(type='password', name='password', required='')
        input(type='submit', value='Submit')
```

Annex 2.4 Exemples de codi del servidor

Aquesta és la aplicació principal, on es gestiona la sessió i es poden veure les principals funcions de middleware amb les seves corresponents rutes d'accés.

app.js

```
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var assert = require('assert');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var session = require('express-session');

var config = require('./config');

var index = require('./routes/index');
var preset = require('./routes/preset');
var users = require('./routes/users');
var login = require('./routes/login');
var register = require('./routes/register');
var debug = require('./routes/debug');

var MongoDBStore = require('connect-mongodb-session')(session);

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');

// uncomment after placing your favicon in /public
//app.use(favicon(path.join(__dirname, 'public', 'favicon.ico')));
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

// Codi de sessió:
var store = new MongoDBStore({
    uri: config.mongoUrl,
    collection: 'sessions'
});
store.on('error', function(error) {
    assert.ifError(error);
    assert.ok(false);
});
app.use(session({
    secret: 'quadramatsecret',
    cookie: {
        maxAge: 1000 * 60 * 60 * 24 * 7 // 1 week
    },
    store: store,
    resave: true,
    saveUninitialized: true
```

```
}));

// Així podem accedir a la session desde totes les vistes,
// sense haver de passar la sessió sempre (al res.render)
app.use(function(req,res,next){
    res.locals.session = req.session;
    next();
});

app.use('/', index);
app.use('/preset', preset);
app.use('/users', users);
app.use('/login', login);
app.use('/register', register);
app.use('/debug', debug);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```

Podem observar com la ruta arrel primer defineix el valor de la variable `loggedIn` en funció de les dades de la sessió i després ens proporciona el contingut de la pàgina `index`.

index.js

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
    var loggedIn = (req.session.email != null && req.session.name != null);

    res.render('index', {
        title: 'Sintetitzador',
        alreadyLoggedIn: loggedIn
    });
});

module.exports = router;
```

La ruta `/register` retorna una pàgina d'error o una de benvinguda en funció del valor de la variable `error` que es passa amb el query. La ruta `/register/new` crea un nou usuari

register.js

```
var express = require('express');
var router = express.Router();
var assert = require('assert');
var mongo = require('mongodb').MongoClient;
var bcrypt = require('bcrypt');

var config = require('../config');

router.get('/', function(req, res, next) {
    var error = (req.query.error == "1");

    res.render('register', {
        title: 'User registration',
        error: error
    });
});

router.get('/welcome', function(req, res, next) {
    res.render('welcome', {
        title: 'Benvingut'
    })
});

/* POST register */
router.post('/new', function(req, res, next) {

    // crear objecte usuari
    var user = {};
    user.email = req.body.email;
    user.nom = req.body.name;
    bcrypt.hash(req.body.password, 10, function(err, hash) {
        // Store hash in database
        user.password = hash;
        // guardar usuari dins mongo
        mongo.connect(config.mongoUrl, function(err, db) {
            assert.equal(null, err);
            var usersCol = db.collection('users');
            usersCol.findOne( { $or : [{ email: user.email}, { nom: user.nom }]
}, function(err, r) {
                assert.equal(null, err);
                if(r) { // el tenim repetit
                    db.close();
                    res.redirect('/register?error=1');
                } else {
                    usersCol.insertOne(user, function(err, r) {
                        assert.equal(null, err);
                        console.log("Usuari inserit amb exit");
                        // sessio nova:
                        req.session.email = req.body.email;
                        req.session.name = req.body.name;
                        db.close();
                        res.redirect('/register/welcome');
                    });
                }
```

```
            });
        });
    });
});


module.exports = router;
```

La ruta `/login` ens permet fer el login de l'usuari a la página `login`, mentre que `/login/logout` estableix els valors de sessió necessaris per tancar-la i ens reenvia a la página principal. La ruta `/login/check` conté el codi per comprovar les dades d'inici de sessió.

login.js

```
var express = require('express');
var router = express.Router();
var assert = require('assert');
var mongo = require('mongodb').MongoClient;
var bcrypt = require('bcrypt');
var config = require('../config');

/* GET home page. */
router.get('/', function(req, res, next) {
    var loggedIn = (req.session.email != null && req.session.name != null);
    var error = (req.query.error == "1");

    res.render('login', {
        title: 'Login',
        alreadyLoggedIn: loggedIn,
        error: error
    });
});

router.get('/logout', function(req, res, next) {
    req.session.email = null;
    req.session.name = null;
    res.redirect('/');
});

router.post('/check', function(req, res, next) {
    // guardar usuari dins mongo
    mongo.connect(config.mongoUrl, function(err, db) {
        assert.equal(null, err);
        var usersCol = db.collection('users');
        usersCol.findOne({ email: req.body.email }, function(err, r) {
            assert.equal(null, err);
            db.close();
            if(r) {
                var email = r.email;
                var name = r.nom;
                var hashedPassword = r.password;
                bcrypt.compare(req.body.password, hashedPassword, function(err,
ok) {
                    if(ok) {
                        // Passwords match
                        req.session.email = email;
                        req.session.name = name;
                        res.redirect('/');
```

```
                    } else {
                        // Passwords don't match
                        res.redirect('/login?error=1');
                    }
                });
            } else { // l'usuari no s'ha logeat correctament
                res.redirect('/login?error=1');
            }
        });
    });
});

module.exports = router;
```

La ruta `/preset/all` ens retorna un objecte JSON amb tots els presets, eliminant el camp email dels presets que no siguin propis. Les rutes `/preset/update`, `/preset/delete` i `/preset/save` actualitzen, esborren i guarden un preset, respectivament.

preset.js

```
var express = require('express');
var router = express.Router();
var assert = require('assert');
var mongo = require('mongodb').MongoClient;
var ObjectID = require('mongodb').ObjectID;

var config = require('../config');

router.get('/all', function(req, res, next) {
    mongo.connect(config.mongoUrl, function(err, db) {
        assert.equal(null, err);
        console.log("Connected successfully to mongodb server.");

        var presetsCol = db.collection('presets');

        presetsCol.find().sort({"autor.nom" : 1, "nom":
1}).toArray(function(err, presets) {
            assert.equal(null, err);
            presets.map(function(p) {
                if(p.autor.email != req.session.email) {
                    delete p.autor.email;
                }
                return p;
            });
            db.close();
            res.json(presets);
        });

    });
});

router.post('/update', function(req, res, next) {
    mongo.connect(config.mongoUrl, function(err, db) {
        assert.equal(null, err);
        console.log("Connected successfully to mongodb server.");

        var presetsCol = db.collection('presets');
        var _id = req.body._id;
        delete req.body._id;
```

```
        presetsCol.updateOne({ _id: ObjectID(_id) }, { $set: req.body },
function(err, result) {
            assert.equal(null, err);
            presetsCol.find().toArray(function(err, presets) {
                assert.equal(null, err);
                db.close();
                res.json(presets);
            });
        });

    });
});

router.post('/delete', function(req, res, next) {
    mongo.connect(config.mongoUrl, function(err, db) {
        assert.equal(null, err);
        console.log("Connected successfully to mongodb server.");

        var presetsCol = db.collection('presets');
        var _id = req.body._id;
        presetsCol.deleteOne({ _id: ObjectID(_id) }, function(err, result) {
            assert.equal(null, err);
            presetsCol.find().toArray(function(err, presets) {
                assert.equal(null, err);
                db.close();
                res.json(presets);
            });
        });

    });
});

router.post('/save', function(req, res, next) {
    mongo.connect(config.mongoUrl, function(err, db) {
        assert.equal(null, err);
        console.log("Connected successfully to mongodb server.");

        var presetsCol = db.collection('presets');

        presetsCol.insertOne(req.body, function(err, result) {
            assert.equal(null, err);
            presetsCol.find().toArray(function(err, presets) {
                assert.equal(null, err);
                db.close();
                res.json(presets);
            });
        });

    });
});

module.exports = router;
```

Annex 3. Estructura de la base de dades

S'han fet servir tres col·leccions, una pels presets amb conjunt de valors del motor d'àudio corresponents als paràmetres modificables per l'usuari dels diferents mòduls (freqüència, tipus de forma d'ona, guanys, etc.) i algunes metadades, una altra pels usuaris i una darrera per emmagatzemar les sessions d'usuari de forma temporal. Com es pot observar en els petits extractes que segueixen, la seva estructura és semblant al JSON (Javascript Object Notation).

Col·lecció "presets" (extracte)


```
{
  "_id": {
    "$oid": "5a2fc1325135d3066f96f680"
  },
  "nom": "Init preset",
  "rating": 0,
  "autor": {
    "nom": "Pere",
    "email": "pereamengual@gmail.com"
  },
  "categoria": "techno",
  "osc1": {
    "oct": 3,
    "semi": 11,
    "cents": 0,
    "wave": "sine",
    "xy": "xy",
    "order": 1,
    "even": false,
    "index": 1
  },
  "osc2": {
    "oct": 3,
    "semi": 11,
    "cents": 0,
    "wave": "sine",
    "xy": "xy",
    "order": 1,
    "even": false,
    "index": 1
  },
  "osc3": {
    "oct": 3,
    "semi": 11,
    "cents": 0,
    "wave": "sine",
    "xy": "xy",
    "order": 1,
    "even": false,
    "index": 1
  },
  "osc4": {
```

```
      "oct": 3,
      "semi": 11,
      "cents": 0,
      "wave": "sine",
      "xy": "xy",
      "order": 1,
      "even": false,
      "index": 1
    },
    "modmatrix": {
      "osc1dest": [20,-36,20,-36,20,-36,20,-36,0.4,-96],
      "osc2dest": [20,-36,20,-36,20,-36,20,-36,0,-96],
      "osc3dest": [20,-36,20,-36,20,-36,20,-36,0,-96],
      "osc4dest": [20,-36,20,-36,20,-36,20,-36,0,-96]
    }
}
{
    "_id": {
      "$oid": "5a3032fb5135d3066f96f683"
    },
    "nom": "morphing flower",
    "rating": 0,
    "autor": {
      "nom": "Pere",
      "email": "pereamengual@gmail.com"
    },
    "categoria": "techno",
    "osc1": {
      "oct": 2,
      "semi": 11,
      "cents": 0.08000000000000007,
      "wave": "sine",
      "xy": "xy",
      "order": 5,
      "even": false,
      "index": 0.8
    },
    "osc2": {
      "oct": 3,
      "semi": 0,
      "cents": -1,
      "wave": "sine",
      "xy": "xy",
      "order": 23,
      "even": false,
      "index": 1.08
    },
    "osc3": {
      "oct": -6,
      "semi": 0,
      "cents": -0.12,
      "wave": "sine",
      "xy": "xy",
      "order": 17,
      "even": false,
      "index": 1.096
    },
    "osc4": {
      "oct": 2,
```

```
      "semi": 11,
      "cents": 0,
      "wave": "square",
      "xy": "xy",
      "order": 9,
      "even": false,
      "index": 1.184
    },
    "modmatrix": {
      "osc1dest": [20,-36,20,-36,20,-36,20,-36,1,-18],
      "osc2dest": [20,-36,20,-36,20,-36,20,-36,0.9749999999999999,-18],
      "osc3dest": [20,-36,20,-12.6,20,-36,20,-10.79,0,-96],
      "osc4dest": [20,-36,20,-36,20,-36,20,-36,0,-96]
    }
  }
  {
    "_id": {
      "$oid": "5a303b9c5135d3066f96f685"
    },
    "nom": "microdots",
    "rating": 0,
    "autor": {
      "nom": "Pere",
      "email": "pereamengual@gmail.com"
    },
    "categoria": "techno",
    "osc1": {
      "oct": 2,
      "semi": 11,
      "cents": -0.040000000000000036,
      "wave": "sine",
      "xy": "xy",
      "order": 1,
      "even": false,
      "index": 1
    },
    "osc2": {
      "oct": 4,
      "semi": 16,
      "cents": 0,
      "wave": "sine",
      "xy": "xy",
      "order": 1,
      "even": false,
      "index": 1
    },
    "osc3": {
      "oct": 6,
      "semi": 20,
      "cents": 0.19999999999999996,
      "wave": "sine",
      "xy": "xy",
      "order": 1,
      "even": false,
      "index": 1
    },
    "osc4": {
      "oct": -6,
      "semi": 0,
```

```
    "cents": 0,
    "wave": "sine",
    "xy": "xy",
    "order": 1,
    "even": false,
    "index": 1
  },
  "modmatrix": {
    "osc1dest": [20,-36,20,-36,20,-36,20,-36,0.675,-18],
    "osc2dest": [20,-36,20,-36,20,-36,20,-36,0.5249999999999999,-18],
    "osc3dest": [20,-36,20,-36,20,-36,20,-36,0.07500000000000001,-18],
    "osc4dest": [20,-36,20,-7.19,20,-11.7,20,-36,0,-96]
  }
}
```


Col·lecció “users” (extracte)


```
{
  "_id": {
    "$oid": "5a3cdfe2fc58ec93f25b5e32"
  },
  "email": "pedrotrotz@gmail.com",
  "nom": "Pedro",
  "password": "$2a$10$9S66llqBN81yUM5BlE5EKubU1JKDht0.zMUDDxXroMWfFr0StDr/C"
}
{
  "_id": {
    "$oid": "5a3ceac98057e69543c7616e"
  },
  "email": "pereamengual@gmail.com",
  "nom": "Pere",
  "password": "$2a$10$Oif4VdjdHXV4mPLZaQsTpe5Uj6/NVdUshGPPKxpuHF1NNeO30eE5C"
}
```

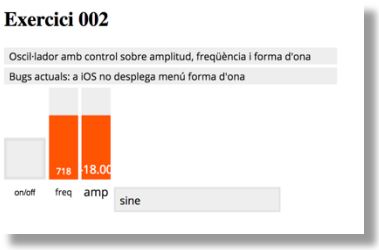

Col·lecció “sessions” (extracte)


```
{
  "_id": "-pzQUHzPDcCYxhI9yhGa0POn3K73VPgT",
  "session": {
    "cookie": {
      "originalMaxAge": 604800000,
      "expires": {
        "$date": "2018-01-13T16:58:09.652Z"
      },
      "secure": null,
      "httpOnly": true,
      "domain": null,
      "path": "/",
      "sameSite": null
    }
  },
  "expires": {
    "$date": "2018-01-13T16:58:09.652Z"
  }
}
```

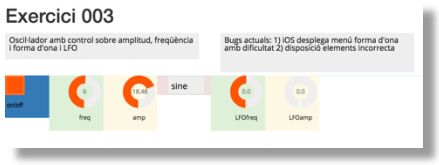
Annex 4. Captures de pantalla

Annex 4.1 Procés d'investigació de les possibilitats de l'API Web Audio.

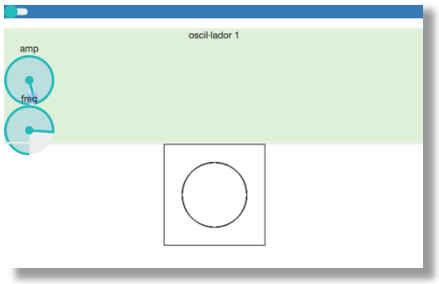
Il·lustració 30 . Exercici 001



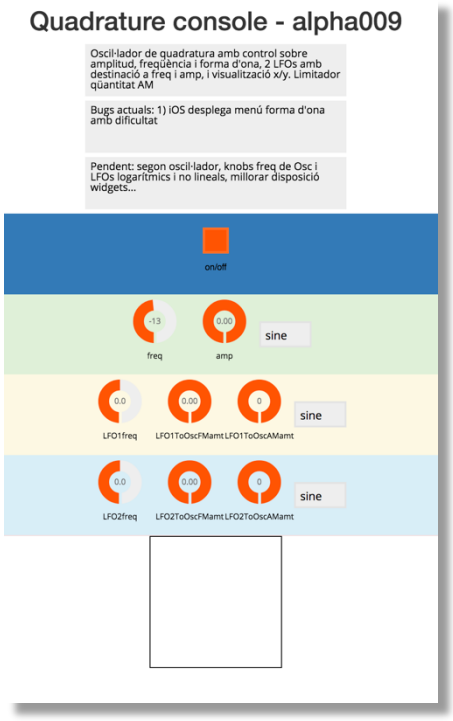
Il·lustració 31 - Exercici 002



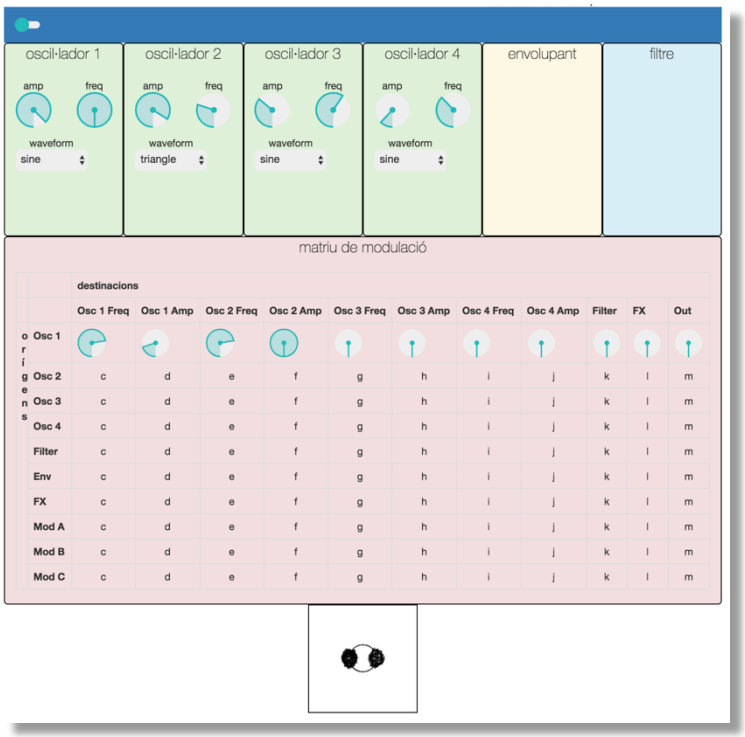
Il·lustració 29 - Exercici 003



Il·lustració 33 . Proves oscil·lador

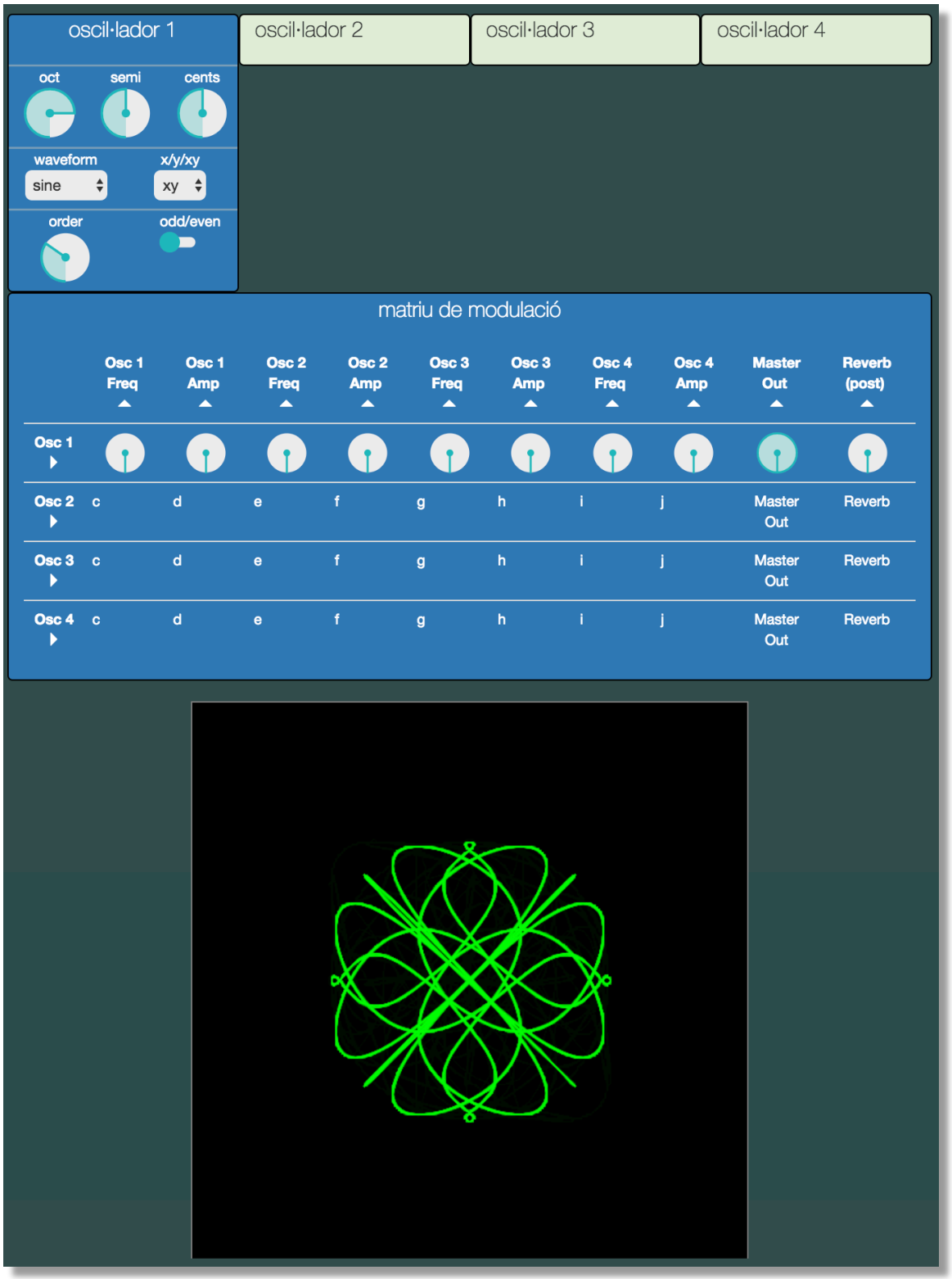


Il·lustració 32 - Proves de modulació



Il·lustració 34 - Proves disposició elements

Annex 4.2 GUI del prototip #1



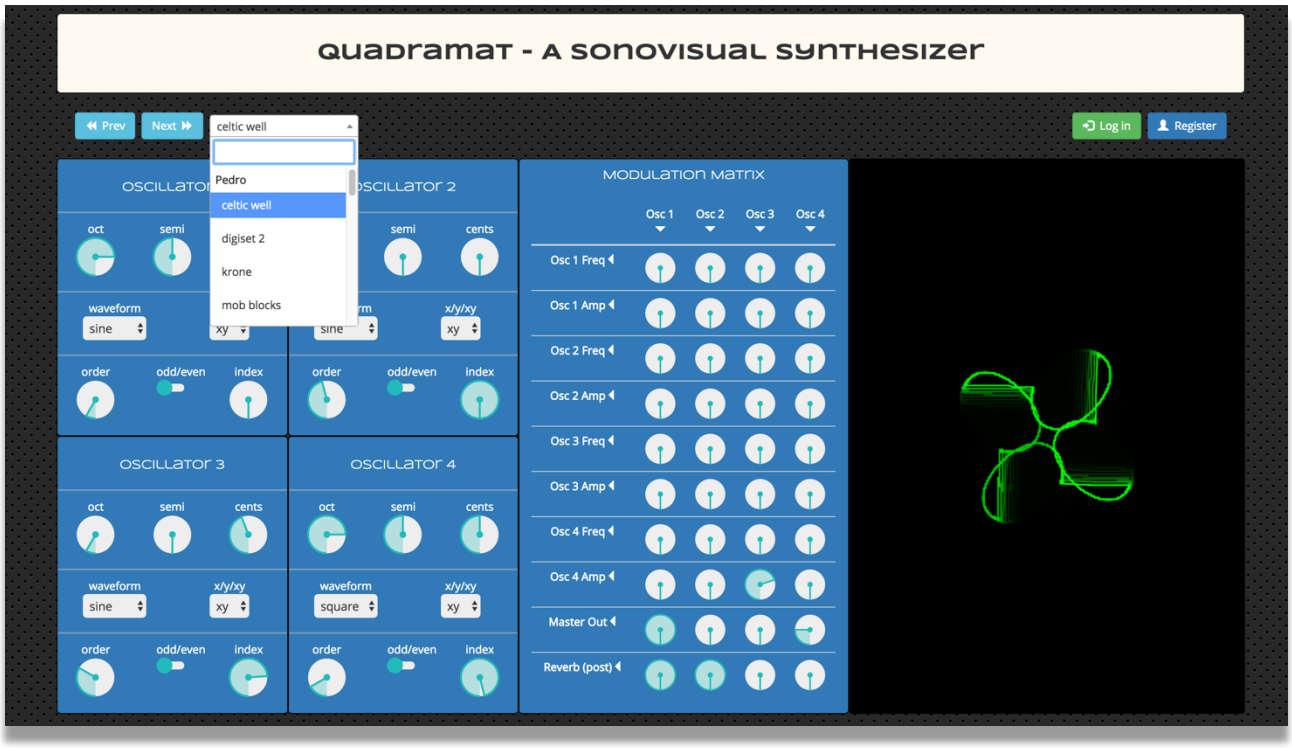
Il·lustració 35 - GUI del prototip #1

Annex 4.3 GUI del prototip #2

Il·lustració 36 - GUI del prototip #2

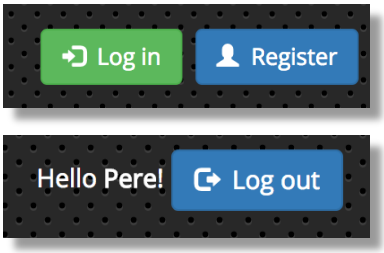
Annex 4.4 GUI del prototip final

El prototip final presenta alguns canvis, sobretot destinats a una millor disposició dels elements en pantalles mòbils. Per exemple, s'ha efectuat una rotació de la matriu de modulació que ara disposa els seus botons amb una orientació vertical.

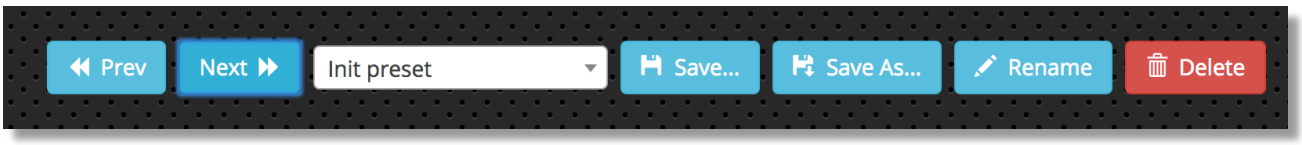


Il·lustració 37 - Pantalla principal GUI definitiva

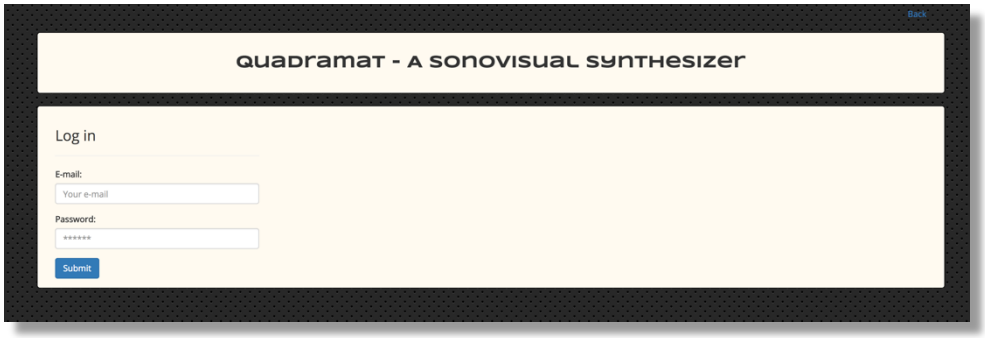
Il·lustració 38 - Detall de l'oscil·lador GUI definitiva



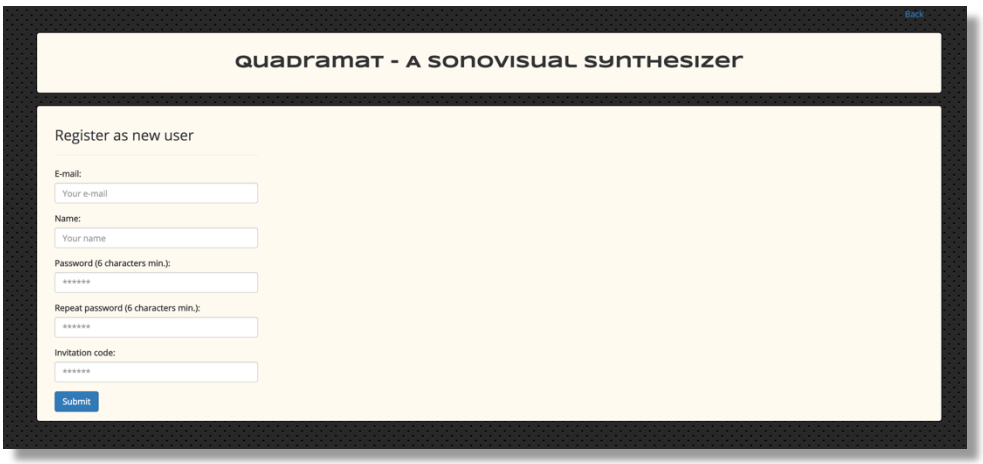
Il·lustració 39 - Botons usuari GUI definitiva



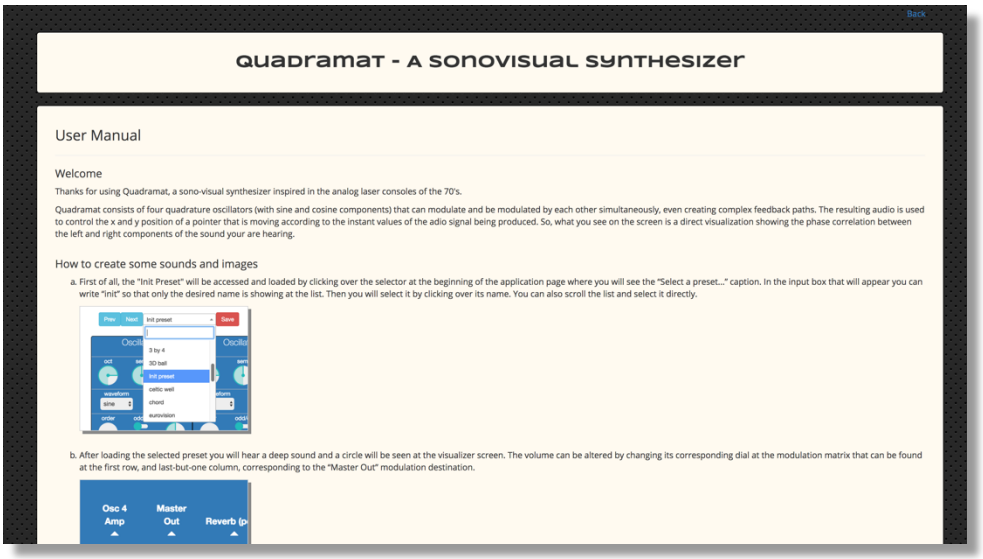
Il·lustració 40 - Botons gestor presets GUI definitiva



Il·lustració 41 - Pàgina Log-in GUI definitiva



Il·lustració 43 - Pàgina Register GUI definitiva



Il·lustració 42 - Pàgina Help GUI definitiva

Annex 5. Guia d'usuari

5.1 Guia d'usuari del prototip #1.

1. Descomprimir l'arxiu zip adjunt

2. Amb el gestor de fitxers del sistema operatiu (Finder per OSX, Explorer per Windows) localitzar dins la carpeta creada el fitxer index.html que es troba dins la carpeta públic

3. Amb un doble clic, obrir la pàgina en el navegador per defecte del sistema operatiu. S'aconsellen Chrome, Safari i Opera. En aquests moments l'aplicació no és compatible amb el navegador Firefox i el prototip actual encara no s'ha provat en Explorer ni Edge

4. Seleccionar els paràmetres oportuns per crear els patrons sono-visual desitjats. A continuació s'indica pas a pas el procediment per crear una figura de baixa complexitat amb només un oscil·lador:

    a. Amb els dials rotatoris "oct", "semi" i "cents" seleccionar la freqüència de l'oscil·lador. Observarem com es pot canviar l'altura del to que s'escolta.

    b. Seleccionar un valor molt baix del dial "oct". Observarem com el desplaçament és tan lent que es pot seguir el moviment del punt de llum a la pantalla i que ara el so té una freqüència tan baixa que no es pot sentir.

    c. Tornem a seleccionar un valor més elevat pel dial "oct" fins que es torni a sentir un so.

    d. Canviem la forma d'ona de l'oscil·lador amb el menú desplegable "Waveform". Observarem que a cada forma d'ona li correspon un timbre sonor i una representació visual diferents. Finalment escollirem "square3", obtenint una ona quadrada amb només els seus primes tres harmònics.

    e. Canviem el valor del dial "order". Observarem com canvia el timbre sonor i també la seva corresponent representació visual. Escollirem una combinació de valor del dial "ordre" i del interruptor "odd/even" que ens resulti interessant.

    f. A la matriu de modulació, canviarem els valors dels dos primers dials, començant per la dreta i per la part superior, i observarem com el primer dial canvia la quantitat de modulació de l'oscil·lador 1 cap a l'oscil·lador 1, és a dir, cap a ell mateix; mentre que el segon dial fa el mateix amb la quantitat de modulació en amplitud.

    g. Podem baixar el volum general amb el penúltim dial, a la columna etiquetada amb el nom "Master Out". Observarem com el volum afecta a les mides de la figura en el visualitzador.

h. Finalment, amb el dial de la columna “Reverb (post)” afegirem la quantitat d’efecte de reverberació desitjada. Observarem que aquest efecte simula la presència de la font sonora dins un espai gran, com una sala de concerts o una església. És aconsellable anar variant els valors dels altres dials per escoltar millor aquest efecte.

5.2 Guia d'usuari del prototip #2.

1. Amb el navegador web, visiteu la pàgina https://frozen-cliffs-27134.herokuapp.com/

2. Seleccioneu els paràmetres oportuns per crear els patrons sono-visual desitjats. A continuació s'indica pas a pas el procediment per crear una figura de complexitat mitja amb dos oscil·ladors:

    a. Per sentir un so, el primer que farem serà pujar el valor del dial de la matriu de modulació de la primera fila, corresponent a Osc1 i a la penúltima columna, corresponent a la sortida "Master Out".

    b. Amb els dials rotatoris "oct", "semi" i "cents" seleccionarem la freqüència de l'oscil·lador. Observarem com es pot canviar l'altura del to que s'escolta.

    c. Seleccionarem un valor molt baix del dial "oct". Observarem com el desplaçament és tan lent que es pot seguir el moviment del punt de llum a la pantalla i que ara el so té una freqüència tan baixa que no es pot sentir.

    d. Tornem a seleccionar un valor més elevat pel dial "oct" fins que es torni a sentir un so.

    e. Canviem la forma d'ona de l'oscil·lador amb el menú desplegable "Waveform". Observarem que a cada forma d'ona li correspon un timbre sonor i una representació visual diferents. Finalment escollirem "square3", obtenint una ona quadrada amb només els seus primes tres harmònics.

    f. Canviem el valor del dial "order". Observarem com canvia el timbre sonor i també la seva corresponent representació visual. Escollirem una combinació de valor del dial "ordre" i del interruptor "odd/even" que ens resulti interessant.

    g. A la matriu de modulació, canviarem els valors dels dos primers dials, començant per la dreta i per la part superior, i observarem com el primer dial canvia la quantitat de modulació de l'oscil·lador 1 cap a l'oscil·lador 1, és a dir, cap a ell mateix; mentre que el segon dial fa el mateix amb la quantitat de modulació en amplitud. De moment, deixarem els dos valors al mínim.

    h. A l'oscil·lador canviarem el valor del paràmetre "oct" per deixar-lo al mínim.

    i. A la matriu de modulació, posarem a la posició de les 3 el dial corresponent a la modulació d'amplitud de l'oscil·lador 2 cap a l'oscil·lador 1; és a dir, el corresponent a la segona fila i la segona columna. Observarem com la figura canvia sola el seu patró i les seves mides mentre que el so presenta una lleugera variació tímbrica de tipus cíclic.

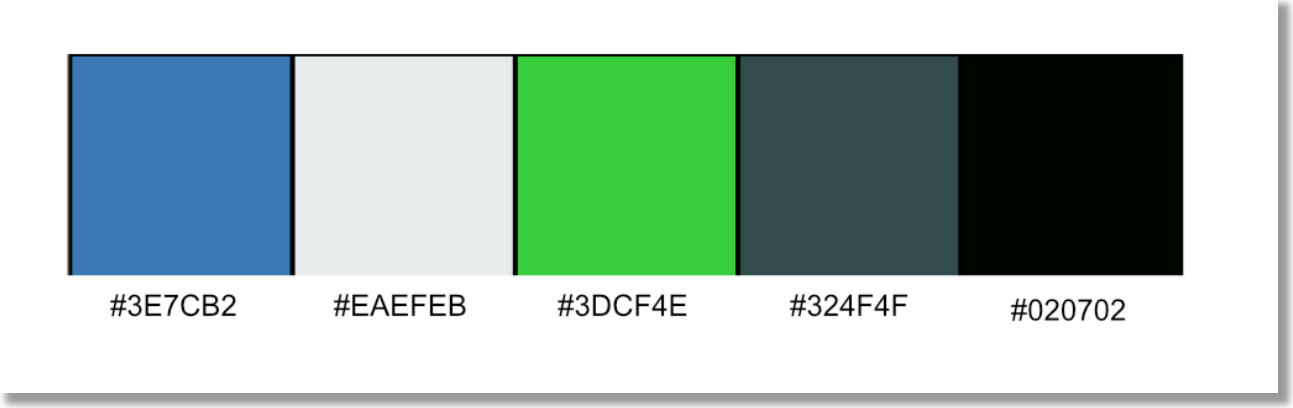
    j. Podem experimentar lliurement amb la resta de paràmetres dels altres oscil·ladors i els valors de la matriu de modulació. Observarem com es poden crear una gran varietat de sons i figures.

k. Podem baixar el volum general de l'oscil·lador 1 amb el penúltim dial de la primera fila, a la columna etiquetada amb el nom "Master Out". Observarem com el volum afecta a les mides de la figura en el visualitzador.

l. Finalment, amb el dial de la columna "Reverb (post)" afegirem a l'oscil·lador 1 la quantitat d'efecte de reverberació desitjada. Observarem que aquest efecte simula la presència de la font sonora dins un espai gran, com una sala de concerts o una església. És aconsellable anar variant els valors dels altres dials per escoltar millor aquest efecte.

Annex 6. Llibre d'estil

Annex 6.1 Paleta de colors



Il·lustració 44 - Paleta de colors

Annex 6.2 Paleta tipogràfica

6.2.1 Títol principal

Quadramat

font-family:"Helvetica Neue", Helvetica, Arial, sans-serif;
font-size:30px;
font-weight:500;

6.2.2 Nom mòdul

Oscil·lador

font-family:"Helvetica Neue", Helvetica, Arial, sans-serif;
font-size:21px;
font-weight:300;

6.2.3 Nom paràmetre

waveform

font-family:"Helvetica Neue", Helvetica, Arial, sans-serif;
font-size:14px;

6.2.4 Nom taula modulació

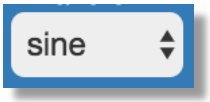
Osc1 Freq

font-family:"Helvetica Neue", Helvetica, Arial, sans-serif;
font-size:14px;
font-weight:bold;

Annex 6.3 Elements de la GUI

Il·lustració 45 - Botó GUI

Il·lustració 47 - Interruptor GUI



Il·lustració 46 - Select GUI

Annex 7. Glossari/Índex analític

Annex 7. Bibliografia

Bibliografia consultada i recursos web

1. **Mozilla.** Web Audio API. *MDN web docs.* [En línia] 2005-2017. [Data: 23 / 10 / 2017.] https://developer.mozilla.org/es/docs/Web_Audio_API.

2. **col·laboradors del projecte Viquipèdia.** Música visual. *Wikipedia.* [En línia] Viquipèdia, l'Enciclopèdia Lliure., 07 / 01 / 2017. [Data: 03 / 10 / 2017.] //ca.wikipedia.org/w/index.php?title=M%C3%BAsica_visual&oldid=18000527.

3. **Systems, Aldebaraan Laser.** Z5 Analog abstract generator. *Aldebaraan Laser Systems.* [En línia] [Data: 03 / 10 / 2017.] http://www.aldebaran-systems.com/Z5.htm.

4. **Earl, David.** A Brief History of FM Synthesis. *A Brief History of FM Synthesis.* [En línia] 01 / 11 / 2013. [Data: 03 / 10 / 2017.] https://ask.audio/articles/a-brief-history-of-fm-synthesis.

5. **Volans, Mo.** Getting to Grips with Modulation Mapping. *tutsplus.* [En línia] 18 / 01 / 2010. [Data: 03 / 10 / 2017.] https://music.tutsplus.com/tutorials/getting-to-grips-with-modulation-mapping--audio-3986.

6. **101010oxo.** Working with the Mini Matrix and the Node Processors. *Rob Hordijk Synths Info Website.* [En línia] 19 / 03 / 2015. [Data: 03 / 10 / 2017.] http://hordijk-synths.info/synth/2015/03/19/minimatrix-node-proc.html.

7. **col·laboradors del projecte Viquipèdia.** Corba de Lissajous. *Wikipedia.* [En línia] Viquipèdia, l'Enciclopèdia Lliure., 22 / 11 / 2016. [Data: 03 / 10 / 2017.] //ca.wikipedia.org/w/index.php?title=Corba_de_Lissajous&oldid=17926034.

8. **X-Laser.** What is ILDA. *X-Laser without limits.* [En línia] 2017. [Data: 25 / 10 / 2017.] http://x-laser.us/faq/what-is-ilda/.

9. **Burucs, Adam.** envatotuts+. *Modular synthesis explained.* [En línia] 15 / 08 / 2016. [Data: 23 / 10 / 2017.] https://music.tutsplus.com/articles/what-is-modular-synthesis--cms-26896.

10. **LXZ Industries.** Cyclops ILDA laser interface. *LXZ Industries.* [En línia] 2017. [Data: 24 / 10 / 2017.] https://www.lzxindustries.net/products/cyclops/.

11. **Cycling '74.** CYCLING ’74: TOOLS FOR SOUND, GRAPHICS, AND INTERACTIVITY. *Max 7.* [En línia] 2017. [Data: 25 / 10 / 2017.] https://cycling74.com/products/max.

12. **Col·laboradors de la Viquipèdia.** Pure data. *Viquipèdia, l'Enciclopèdia Lliure.* [En línia] 2017. [Data: 25 / 10 / 2017.] //ca.wikipedia.org/w/index.php?title=Pure_data&oldid=18252100.

13. **Enzien Audio.** What is Heavy? *Heavy Audio Tools.* [En línia] 2017. [Data: 25 / 10 / 2017.] https://enzienaudio.com/docs/index.html#01.introduction#what-is-heavy-.

14. **Apple Computer.** Introduction to Quartz Composer User Guide. *Developer's site.* [En línia] 2007. [Data: 25 / 10 / 2017.] https://developer.apple.com/library/content/documentation/GraphicsImaging/Conceptual/QuartzComposerUserGuide/qc_intro/qc_intro.html.

15. **joreg.** vvvv. *vvvv.* [En línia] 2017. [Data: 25 / 10 / 2017.] https://vvvv.org/.

16. **altres, James McCartney i.** Supercollider. *Welcome to Supercollider.* [En línia] 2017. [Data: 25 / 10 / 2017.] https://github.com/supercollider/supercollider.

17. **OpenFrameworks.** OpenFrameworks. *About OpenFameworks.* [En línia] 2017. [Data: 25 / 10 / 2017.] http://openframeworks.cc/about/.

18. **Grierson, Mick.** Maximilian, a cross platform c++ audio synthesis library for artists learning to program. *Goldmiths University of London. Department of Computing.* [En línia] Gener / 2010. [Data: 25 / 10 / 2017.] http://www.doc.gold.ac.uk/~mas01sd/papers/internalReports/grierson1.pdf.

19. **Processing foundation.** Processing. *Processing.* [En línia] 2017. [Data: 25 / 10 / 2017.] https://processing.org/.

20. **Smus, Boris.** Web Audio API. *O'Reilly Atlas.* [En línia] 2013. [Data: 25 / 10 / 2017.] http://chimera.labs.oreilly.com/books/1234000001552.

21. **Processing foundation.** p5.js. *p5.js.* [En línia] 2017. [Data: 25 / 10 / 2017.] https://p5js.org/.

22. **gitHub.** Charlie Roberts' gitHub. *gibber.js. Music and audio programming for p5.js.* [En línia] 2017. [Data: 25 / 10 / 2017.] https://github.com/charlieroberts/p5.gibber.js/tree/master.

23. **Burk/Polansky/Repetto/Roberts/Rockmore.** Chapter 4: The Synthesis of Sound by Computer Section 4.6: Waveshaping. *Music and Computers. A theoretical and historical approach.* [En línia] [Data: 23 / 10 / 2017.] http://cmc.music.columbia.edu/musicandcomputers/chapter4/04_06.php.

24. **Wikipedia contributors.** Software development process. *Wikipedia.* [En línia] 03 / Octubre / 2017. https://en.wikipedia.org/w/index.php?title=Software_development_process&oldid=803258637.

25. **col·laboradors del projecte Viquipèdia.** Model de prototips. *Wikipedia.* [En línia] Viquipèdia, l'Enciclopèdia Lliure., 10 / 06 / 2017. [Data: 03 / 10 / 2017.] //ca.wikipedia.org/w/index.php?title=Model_de_prototips&oldid=18530094.

26. **Studytonight.com.** Mongo. *Studytonight.* [En línia] 2018. [Data: 05 / 01 / 2018.] http://www.studytonight.com/mongodb/mongodb-vs-rdbms.

27. **Finley, Klint.** TechCrunch. *https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/.* [En línia] 14 / 07 / 2012. [Data: 29 / 10 / 2017.] https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/.

28. **Man, Yotam.** Tone.js types r11. *Tone.js docs.* [En línia] 2017. [Data: 20 / 10 / 2017.] https://tonejs.github.io/docs/r11/Type.

29. —. Tone.js connections. *Tone.js docs.* [En línia] 2017. [Data: 23 / 10 / 2017.] https://github.com/Tonejs/Tone.js/wiki/Connections .

30. **Wikipedia contributors.** Pythagorean tuning. *Wikipedia.* [En línia] Wikipedia, The Free Encyclopedia, 24 / 10 / 2017. [Data: 30 / 10 / 2017.] https://en.wikipedia.org/wiki/Pythagorean_tuning.

31. **Autors de wikibooks.** Sound Synthesis Theory/Modulation Synthesis. *Sound Synthesis Theory.* [En línia] 07 / 04 / 2011. [Data: 25 / 10 / 2017.] https://en.wikibooks.org/wiki/Sound_Synthesis_Theory/Modulation_Synthesis .

32. **Osmani, Addy.** Learning Javascript Design Patterns. *Addy Osmani. Engineering manager at Google working on Chrome.* [En línia] O' Reilly, 2017. [Data: 23 / 10 / 2017.] https://addyosmani.com/resources/essentialjsdesignpatterns/book/.

33. **Kasireddy, Preethi.** JavaScript Modules: A Beginner's Guide. *freeCodeCamp.* [En línia] 22 / 01 / 2016. [Data: 15 / 11 / 2017.] https://medium.freecodecamp.org/javascript-modules-a-beginner-s-guide-783f7d7a5fcc.

34. **Karanth, Deepak.** Is your code DRY or WET? *Software yoga.* [En línia] 26 / 10 / 2015. [Data: 15 / 11 / 2017.] https://www.softwareyoga.com/is-your-code-dry-or-wet/.

35. **StrongLoop, IBM, and other expressjs.com contributors.** Installing. *Express.* [En línia] 2017. [Data: 15 / 11 / 2017.] http://expressjs.com/en/starter/installing.html.

36. **StrongLoop, IBM, and other expressjs.com contributors.** Express aplication generator. *Express.* [En línia] 2017. [Data: 15 / 11 / 2017.] http://expressjs.com/en/starter/generator.html.

37. **Salesforce.com.** Heroku Dev Center. *mLab MongoDB.* [En línia] 11 / 10 / 2017. [Data: 1 / 12 / 2017.] https://devcenter.heroku.com/articles/mongolab.

38. **MongoDB Inc.** The Mongo Shell. *MongoDB Documentation.* [En línia] 2008. [Data: 1 / 12 / 2017.] https://docs.mongodb.com/v3.2/mongo/.

39. **mongoDB Inc.** Drivers. *MongoDB Documentation.* [En línia] 2008. [Data: 1 / 12 / 2017.] https://docs.mongodb.com/getting-started/shell/drivers/.

40. **MongoDB Inc.** MongoDB Node.js Driver. *MongoDB Documentation.* [En línia] 2008. [Data: 1 / 12 / 2017.] http://mongodb.github.io/node-mongodb-native/2.2/tutorials/connect/.

41. **ICOconvert.com.** ICO convert. *ICO convert.* [En línia] 2017. [Data: 1 / 12 / 2017.] http://icoconvert.com/.

42. **Affordable Usability.** Memorability | Can Visitors Repeat Tasks Search . *Affordable usability web design.* [En línia] 2011. [Data: 15 / 12 / 2017.] http://www.affordableusability.com/usability/memorability.html.

43. **ExpressJs.** Express-session. *Express-session.* [En línia] [Data: 27 / 12 / 2017.] https://github.com/expressjs/session.

44. **npm.** bcrypt. *nom packages.* [En línia] [Data: 30 / 12 / 2017.] https://www.npmjs.com/package/bcrypt.

45. **Wikipedia contributors.** HTML5 Audio. *Wikipedia, The Free Encyclopedia.* [En línia] 18 / 10 / 2017. [Data: 30 / 10 / 2017.] https://en.wikipedia.org/w/index.php?title=HTML5_Audio&oldid=805862173.

46. **Mozilla i contribuents individuals.** Web Audio API. *MDN web docs.* [En línia] 2055-2017. [Data: 25 / 10 / 2017.] https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API.

47. **Mozilla and individual contributors.** Canvas API. *MDN web docs.* [En línia] 2005-2017. [Data: 30 / 10 / 2017.] https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API.

48. **Man, Yotam.** StartAudioContext. *GitHub.* [En línia] 2016. [Data: 30 / 10 / 2017.] https://github.com/tambien/StartAudioContext.

49. **Apple Inc.** Playing Sounds with the Web Audio API. *Safari HTML5 Audio and Video Guide.* [En línia] 13 / 12 / 2012. [Data: 30 / 10 / 2017.] https://developer.apple.com/library/content/documentation/AudioVideo/Conceptual/Using_HTML5_Audio_Video/PlayingandSynthesizingSounds/PlayingandSynthesizingSounds.html.

50. **core team and contributors.** Bootstrap. *Bootstrap.* [En línia] [Data: 30 / 10 / 2017.] http://getbootstrap.com/.

51. **Taylor, Ben.** NexusUI 2.0 Web Audio Interfaces. *NexusUI 2.0.* [En línia] 2017. [Data: 30 / 10 / 2017.] https://nexus-js.github.io/ui/.

52. **Weissman, Alexander.** Select2. *Select2.* [En línia] 2017. [Data: 15 / 12 / 2017.] https://select2.org/.

53. **Stack Exchange Inc.** Stack Overflow. *What does body-parser do with express? .* [En línia] 2017. [Data: 30 / 12 / 2017.] https://stackoverflow.com/questions/38306569/what-does-body-parser-do-with-express.

54. **GitHub, Inc.** Lightweight MongoDB-backed session store for Connect and Express. *GitHub.* [En línia] 2017. [Data: 30 / 12 / 2017.] https://github.com/mongodb-js/connect-mongodb-session.

55. **Codementior.** Cookie management in Express. *Codementor.* [En línia] 2017. [Data: 30 / 12 / 2017.] https://www.codementor.io/noddy/cookie-management-in-express-js-du107rmna.

56. **GitHub.** Express-favicon. *GitHub.* [En línia] 2017. [Data: 30 / 12 / 2017.] https://github.com/expressjs/serve-favicon.

57. **Nodewebapps.** How do NodeJs sessions work. *Nodewebapps.* [En línia] 2017. [Data: 31 / 12 / 2017.] https://nodewebapps.com/2017/06/18/how-do-nodejs-sessions-work/.

58. **npm.** MongoDB. *npm.* [En línia] 2017. [Data: 31 / 12 / 2017.] https://www.npmjs.com/package/mongodb.

59. —. npm. *npm pug.* [En línia] 2017. [Data: 31 / 12 / 2017.] https://www.npmjs.com/package/pug.

60. **Cympanu, Catalin.** https://www.bleepingcomputer.com/news/security/mongodb-apocalypse-is-here-as-ransom-attacks-hit-10-000-servers/. *bleeping computer.* [En línia] 07 / 01 / 2017. [Data: 30 / 10 / 2017.] https://www.bleepingcomputer.com/news/security/mongodb-apocalypse-is-here-as-ransom-attacks-hit-10-000-servers/.

61. **Ottenheimer, Davi.** Update: How to Avoid a Malicious Attack That Ransoms Your Data. *mongoDB for giant ideas.* [En línia] 08 / 09 / 2047. [Data: 30 / 10 / 2017.] https://www.mongodb.com/blog/post/update-how-to-avoid-a-malicious-attack-that-ransoms-your-data.

62. **Prescott, Susan.** The top 10 web application security risks. *AT&T bussiness.* [En línia] [Data: 30 / 10 / 2017.] https://www.business.att.com/learn/operational-effectiveness/the-top-10-web-application-security-risks.html.

63. **US Department of Health & Human Services.** Usability Testing. *usabilty.gov.* [En línia] 01 / 11 / 2017. [Data: 01 / 11 / 2017.] https://www.usability.gov/how-to-and-tools/methods/usability-testing.html.

64. **Cad, Jerry.** User testing, explained. *TNW.* [En línia] 24 / 04 / 2015. [Data: 30 / 10 / 2017.] https://thenextweb.com/creativity/2015/04/27/user-testing-explained/.

65. **Wikipedia contributors.** HTML sanitization. *Wikipedia.* [En línia] 23 / 12 / 2017. [Data: 31 / 12 / 2017.] https://en.wikipedia.org/w/index.php?title=HTML_sanitization&oldid=816756503.

66. **Defuse security.** Salted Password Hashing - Doing it Right. *crackstation.* [En línia] 1 / 8 / 2017. [Data: 30 / 12 / 2017.] https://crackstation.net/hashing-security.htm#normalhashing.

67. **Google.** Download and install google chrome. *Google.* [En línia] 2017. [Data: 31 / 12 / 2017.] https://support.google.com/chrome/answer/95346?hl=en&rd=1.

68. **LXZ Industries.** Cyclops. *LXZ Industries.* [En línia] 2017. [Data: 31 / 12 / 2017.] https://www.lzxindustries.net/products/cyclops/.

69. **Sweetwater Inc.** Sweetwater InSync. *Sweetwater.* [En línia] 2017. [Data: 30 / 12 / 2017.] https://www.sweetwater.com/insync/mod-wheel/.

70. **Sweetwater.** InSync. *Sweetwater.* [En línia] 2017. [Data: 30 / 12 / 2017.] https://www.sweetwater.com/insync/ribbon-controller/.

71. **Skylab coders academy.** ¿Cuál es el salario de un programador? *Skylab coders academy.* [En línia] 4 / 10 / 2016. [Data: 30 / 10 / 2017.] http://www.skylabcoders.com/es/-cu%C3%A1l-es-el-salario-de-un-programador-_13037.

72. **Marc.** Guía salarial para programadores: Octubre 2015 (Barcelona). *Jobfluent.* [En línia] 1 / 10 / 2015. [Data: 30 / 10 / 2017.] https://www.jobfluent.com/blog/guia-salarial-para-programadores-barcelona-2015?lang=es.

73. **Free Software Foundation.** What is free software. *Free software foundation.* [En línia] 2004-2017. [Data: 30 / 10 / 2017.] http://www.fsf.org/about/what-is-free-software.

74. **Wikipedia contributors.** The Free Software Definition. *Publisher: .* [En línia] 03 / 09 / 2017. [Data: 30 / 10 / 2017.] https://en.wikipedia.org/w/index.php?title=The_Free_Software_Definition&oldid=798753665.

75. **Free Software Foundation.** Llista de llicències i comentaris sobre elles. *El sistema operatiu GNU.* [En línia] 28 / 06 / 2000. [Data: 30 / 10 / 2010.] http://www.gnu.org/licenses/license-list.html.

76. **Drake, mark.** The Difference Between Free and Open-Source Software. *Digital Ocean.* [En línia] 30 / 10 / 2017. [Data: 01 / 11 / 2017.] https://www.digitalocean.com/community/tutorials/Free-vs-Open-Source-Software.

77. **open source initiative.** Licenses & Standards. *open source initiative.* [En línia] [Data: 30 / 10 / 2017.] https://opensource.org/licenses.

78. **Amengual-Gomila, Pere.** License. *Github.* [En línia] 01 / 11 / 2017. [Data: 01 / 11 / 2017.] https://github.com/pereamengual/QuadraMat/blob/master/LICENSE.

79. **MongoDB, Inc.** Install MongoDB Community Edition on macOS. *MONGODB MANUAL.* [En línia] 2008. [Data: 1 / 12 / 2017.] https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/.

Copyright imatges

Persona 2: Best-Antiaging-Skin-Care-Women-Over-30 Popsugar © Benjamin Stone [en línia]. Disponible a https://www.popsugar.com/beauty/Best-Antiaging-Skin-Care-Women-Over-30-41089056

Annex 8. Vita

Pere Amengual, conegut també amb el sobrenom de *Trotz*, és un artista electrònic que va enregistrar els seus primers treballs de música electrònica experimental a mitjans dels anys 80, en un estil entre l'industrial i el *cut-up*. Ha estudiat, entre d'altres, el *Grau en Multimèdia* a la Universitat Oberta de Catalunya, *Sistemes d'àudio i sonorització* a la Universitat de València, *Disseny d'espais interactius* a la School of Machines (Berlin) i *Max/Msp/Jitter* amb Yaciane Setbi i Julien Bayle. Als anys 90 va col·laborar en l'enregistrament d'alguns dels primers èxits de l'escena House de Mallorca i va contribuir a difondre la música electrònica com membre de col·lectius com *Tecnofamilia*, *Roñatrón* i, més tard, els segells *Virus/Antivirus*. En els seus darrers treballs ha abandonat els ritmes propis de la pista de ball per tornar al renou i la creació sonora més experimental, incorporant elements d'interacció i art sono-visual de creació pròpia.