Eines digitals per a la producció musical clàssica

Desenvolupament d'una apliacacció per a protocols d'enregistrament

Per:

ADRIÀ MARTÍNEZ I CAZORLA

Department de Sonologia
ESCOLA SUPERIOR DE MÚSICA DE CATALUNYA

TUTOR: ENRIC GUAUS

MAIG 2017

Abstracte

El treball consistirà en el desenvolupament d'una aplicació per a tauleta que sigui d'ajuda a la feina del productor musical. La intenció és crear una aplicació amb la qual es puguin fer els protocols d'enregistrament durant la sessió de gravació i que tingui funcions d'ajuda al muntatge com suggerir possibles preses bones, talls, etc. En aquesta memòria es pot veure un resum de tot el procediment des de la idea inicial, passant pel disseny funcional, la implementació del prototip i uns petits apunts del que caldrà fer abans que surti al mercat.

El trabajo consistirá en el desarrollo de una aplicación para tableta que sea de ayuda en la tarea del productor musical. La intención es crear una aplicación con la que se puedan hacer los protocolos de grabación durante las sesiones y despues tenga funciones de ayuda en el montaje como sugerir posibles tomas buenas, cortes, etc. En ésta memoria se puede ver todo el procedimiento des de la idea inicial, pasando por el diseño funcional, la implementación del prototipo y unos pequeños apuntes del que se deberá hacer para que salga al mercado.

The purpose of the project is to develop an iPad app that will help producers to do their job more easily. The app will help producers to write 'take based notes' while recording and to use these notes during the post-production process, suggesting the best takes, cuts, etcetera. The thesis describe the process since the first idea. It include the functional design, prototype implementation and a brief description of the next steps needed before the app being released.

Dedicació i agraiments

A tota aquella gent que m'ha ajudat a realitzar aquest treball ja sigui en qüestió d'informació i consells o en la part emocional donant-me suport en els dies que les coses no funcionaven com un desitjava.

En especial, cal remarcar l'ajuda que he obtingut del meu tutor, l'Enric Guaus, del Pere Casulleras i d'altres companys i professors de l'ESMUC. També m'agradaria donar les gràcies al Joan García, qui m'ha donat un cop de ma en els aspectes més tècnics de Swift i que sense aquesta empenta no hagués pogut arribar fins on he arribat en el desenvolupament del prototip.

TAULA DE CONTINGUTS

CAPÍTOL 1

INTRODUCCIÓ

1.1 Motivació a fer aquest treball

En l'àmbit de la sonologia, el treball de final de grau pot tenir molts vessants i com a mostra, cada any hi ha hagut treballs ben diversos. Des d'un treball estrictament teòric fins a algun altre que la part teòrica és anecdòtica i merament una formalitat. Potser el més comú, i no per això menys interessant, és el de dur a terme una producció. Sigui la producció i/ o enregistrament d'una maqueta, un disc, un concert, una composició pròpia utilitzant els recursos sonològics pertinents, o bé una barreja de les anteriors. Encara que la meva intenció un cop acabi la carrera, i que a hores d'ara ja estic duent a terme, és la de treballar en aquest àmbit (el de la producció musical), el fet de fer un treball sobre això mai ha estat la meva prioritat.

Des d'un primer moment la intenció ha estat fer un treball de final de grau del qual se'n pogués treure un rendiment en el futur laboral, però que no fos estrictament "el mateix"que acabaria fent. En altres paraules, volia fer una eina útil relacionada amb la producció musical, però no una producció musical per se.

No penso que no sigui interessant fer un treball sobre com has dut a terme una producció, estic segur que s'aprèn molt, però si realment vull acabar-me dedicant a això, ja tindré molt de temps per aprendre i fer produccions musicals. Si tot va bé i tinc feina, en canvi, potser no tindré el temps necessari per a desenvolupar aquestes "eines"o ütilitatsï és per això que la intenció ha estat fer-ho ara, aprofitant com a excusa el treball de final de grau.

Arribats a aquest punt, l'opció que em cridava més l'atenció era la de fer alguna cosa relacionada amb la programació. Al llarg de la carrera, ha estat un tema que m'ha atret molt i he tingut la facilitat i l'interès per treure-li tot el rendiment que he pogut, tant en les tres assignatures de programació com en d'altres on la programació ha estat un mitjà. És per això que no volia desaprofitar aquest esforç fet durant aquests quatre anys i acabar d'esprémer la programació portant el TFG cap a aquest àmbit.

Havent arribat a aquestes conclusions, les opcions de possibles treballs s'anaven reduint. Havia de ser una eina informàtica que fos útil en el terreny de la producció musical. En aquest cas i per acotar encara més, de la producció musical de la música clàssica.

Actualment es podria dir que està gairebé tot inventat i personalment no em despertava massa interès intentar fer un treball sobre alguna cosa que ja està inventada tornant-la a fer, com per exemple seria intentar fer un programa d'edició d'àudio o bé un plug-in d'efectes bàsic. Avui dia hi ha un munt d'empreses especialitzades en aquest sector que tenen tot un equip de gent molt més capacitada que jo, a banda que un projecte amb cara i ulls és massa gran i ambiciós per a ser un treball de final de grau. Per tant, aquesta opció va ser descartada des d'un inici. Hi havia una cosa en l'àmbit de la producció musical de música clàssica que encara tothom feia manualment, amb llapis, goma, i partitures en paper. Es tracta dels protocols d'enregistrament, la gent ho fa en paper bàsicament perquè encara no hi ha cap aplicació que els hi permeti fer-ho informàticament.

El protocol d'enregistrament són totes aquelles anotacions que es fan sobre la partitura abans, durant i després d'un enregistrament de música (majoritàriament) clàssica. S'entrarà en més detall de què és i perquè serveix aquest protocol a la següent secció però el que interessa ara és que per primer cop, havia trobat un forat on encabir el treball.

La intenció, doncs, va ser la de fer una aplicació per a tauleta que permetés dur a terme aquest protocol d'enregistrament i treballar amb les dades que s'hi anessin anotant per facilitar la feina a la postproducció.

1.2 Inspiració

1.2.1 Què és el protocol d'enregistrament?

Abans d'entrar en detall sobre com s'ha plantejat i desenvolupat l'app, s'ha cregut convenient fer una petita recerca sobre què és això del protocol d'enregistrament, per a què serveix i quan s'utilitza.

En Pere Casulleras en els seus apunts de l'assignatura Tècniques d'enregistrament i postproducció I[1] ho defineix d'aquesta manera:

Durant un enregistrament normalment es fan diverses versions de les peces, per a tenir la possibilitat d'escollir la millor o de combinar-ne vàries per a aconseguir una versió final muntada. Les estratègies per a enregistrar peces de música són diverses i depenen de molts factors, però en qualsevol cas necessitem un sistema de notació que ens permeti per una part de tenir un control detallat de tot el que va succeint i per altra part en permeti de tenir una visió global de la qualitat de cada passatge, sense que ens hi perdem.

I el més important: *Un bon protocol ens ha de poder informar ràpidament de la qualitat de les versions individuals en cada lloc concret i també de si disposem de prou*

[1] `http://www.casacota.net/perl?num=1129340876`

material de qualitat a tot arreu, de manera que a l'hora de fer un pla de muntatge l'elecció de les versions es pugui fer de manera inequívoca.

Actualment, en els protocols *a paper* si no s'és molt ordenat i/o es fan moltíssimes preses per cada secció, poder tenir aquesta visió global de la qual parla en Pere Casulleras és realment complicat. Per altra banda, en una partitura d'orquestra per exemple, una sola presa pot durar 10-15 pàgines o més, així que per saber en un compàs concret quines preses hi ha hagut és una feina bastant feixuga de pàgines amunt i pàgines avall. Tots aquests aspectes són els que s'intentaran simplificar amb el desenvolupament de l'aplicació.

Els apunts de en Pere Casulleras, són l'única informació escrita trobada parlant sobre el protocol d'enregistrament. S'ha fet una recerca tant per internet com en la poca bibliografia especialitzada que s'ha trobat i enlloc apareix cap explicació de què és ni com s'utilitza el protocol d'enregistrament. En la majoria de publicacions consultades com el llibre de Hepworth-Sawyer and Golding (2011) fan referència a la importància de ser ordenat i anotar tot el que va passant durant l'enregistrament, però en cap cas anomenen el protocol o alguna cosa similar.

Aquest fet es pot deure al fet que cada productor o cada escola de productors fa servir la metodologia que li va millor per anotar el que passa a cada presa. El que sí que és clar, és que tots tenen un element en comú i aquest és la partitura i les anotacions *per presa*. Per tant, es pot arribar a dues conclusions prèvies a desenvolupar l'aplicació: La primera és que l'app haurà de ser prou personalitzable com per poder arribar al màxim de públic possible i la segona és que ha d'estar basada sobre un visor de partitures i el número de presa.

Per poder arribar al màxim de gent possible i poder desenvolupar l'aplicació d'una manera més *universal*, s'ha enviat una enquesta (de la que se'n parlarà al capítol 2 *Treball de camp*) a tots els productors, tonmeisters i estudiants s'han pogut trobar per a veure com és la visió d'aquest protocol aquí i a l'estranger. Potser ara mateix no es farà servir per a desenvolupar el prototip de l'aplicació, ja que pot ser molt complicat encabir-hi totes les visions en un treball, però el que és segur és que per a futures actualitzacions serà de molta utilitat.

1.2.2 Altres aplicacions similars al mercat

En els principals mercats d'aplicacions com són l'AppStore[2] i el GooglePlay[3], existeixen infinitat d'aplicacions que permeten llegir arxius en format PDF. Potser la més comuna en aquest camp és la pròpia d'Adobe: L'Adobe Acrobat Reader,[4] però també s'hi poden trobar aplicacions que estan creades específicament com a lectors de partitures. Per exemple, l'anomenada ForScore[5] és la que fan servir més músics actualment, ja que està molt ben aconseguida i ofereix moltes eines útils per a fer anotacions relacionades amb la música, com ara crescendos/diminuendos, altres signes d'expressió, dinàmiques, etc... Totes dues apps (i moltes d'altres que s'han provat abans de fer el treball) a banda de veure la partitura en sí, permeten fer anotacions utilitzant la pantalla tàctil dels dispositius, tot i així, perquè siguin precises i no ocupin mitja pantalla es necessita fer zoom abans d'escriure i es perd molt de temps fent zoom amunt i zoom avall. Per altra banda, per escriure-hi s'ha de deixar premuda la pantalla durant un segon o bé prémer un botó que permeti entrar en el mode escriptura. El problema en aquest cas és que per passar de pàgina s'ha de sortir del mode d'edició i després tornar-hi a entrar un cop s'ha passat pàgina.

Tots aquests handicaps fan que el fet d'escriure un protocol d'enregistrament amb alguna d'aquestes apps sigui bastant feixuc, ja que per qualsevol anotació a fer, es perd molt de temps per tots cantons. No és impossible, s'ha provat i s'ha pogut acabar fent, però tampoc és gens pràctic. Quan es fa un protocol d'enregistrament, el temps passa amb la música i no es pot estar *perdent el temps* tocant el zoom o bé passant pàgina

[2] https://itunes.apple.com/es/store
[3] https://play.google.com/store
[4] https://acrobat.adobe.com/la/es/acrobat/pdf-reader.html
[5] https://forscore.co/

d'una manera massa complicada. Possiblement per això, encara ningú fa els protocols utilitzant una tauleta.

1.3 Per què Apple?

La primera pregunta plantejada abans de començar a desenvolupar l'app va ser: "Per a quin sistema operatiu s'ha de programar? iOS? Android? Windows?"Després d'una recerca per Internet sobre els pros i contres de cadascun dels sistemes operatius, es va acabar triant Apple (i per tant iPad) bàsicament per la seva facilitat de programació i menor "manteniment"de l'app (un cop publicada) respecte als altres.

Apple de per si és molt tancat i això té desavantatges però també té la seva part bona. D'iPad ara mateix només hi ha 3 models (iPad mini, iPad air i iPad pro) en canvi, de tauletes Android o Windows existeixen centenars de models diferents. Només per aquest simple fet, ja es pot deduir que serà més fàcil desenvolupar una aplicació que funcioni bé amb tres models que no pas una que hagi de funcionar amb tota la resta. Però no només això, els dispositius que porten sistema operatiu Android, provenen de diversos fabricants com poden ser Samsung, Asus, Sony, entre molts d'altres, per tant, cadascun tindrà les seves característiques i encara que utilitzin el mateix sistema operatiu, serà més complicat programar-ho de tal manera perquè tots facin el mateix.

Això també és un avantatge per Apple, ja que el llenguatge de programació utilitzat està creat exclusivament per aquest tipus de dispositiu i es té accés a moltes funcions de manera més directe. Per altra banda, el llenguatge Swift, del que se'n parlarà més endavant al capítol 4, està basat en el llenguatge C o C++ que és el que s'aprèn a l'Esmuc. Per acabar, un altre factor a tenir en compte és que el fet que hi hagi menys dispositius, també fa que hi hagi menys actualitzacions del sistema i per tant, no s'ha d'estar tan pendent de canviar el codi perquè funcioni a cada actualització més que unes poques vegades a l'any. Si no s'és programador professional, com és el cas, això es converteix en un gran avantatge.

1.4 Plantejament

La idea començava a estar clara, però encara s'havia de decidir per on començar. Fer una aplicació de zero és un projecte bastant ambiciós i si no està ben plantejat des d'un bon començament, pot ser un fracàs. Primer de tot, es va traslladar la idea d'app en un esquema per tal de fixar les idees inicials, el simple fet d'escriure-les i haver d'esquematitzar-ho va fer veure quins aspectes es tenien clars sobre el funcionament de l'app i quins altres calia acabar de polir. Després d'aquest primer esquema i de fer un mockup del funcionament bàsic de l'aplicació, es va començar a programar com a mínim fins a un cert punt on es pogués veure una mostra del funcionament i les possibilitats, és a dir, un prototip. Arribats a aquest punt, comença el treball en paral·lel de programació i d'escriptura de la memòria.

La feina de programació va començar molt abans que de la d'escriptura de la memòria, ja que requereix més temps, és per això que probablement s'explicaran coses en algun apartat d'aquest treball que seran incoherents amb d'altres que s'han explicat anteriorment. Això és degut al fet que a mesura que ha anat evolucionant el projecte, s'ha anat redissenyant sigui per causes d'implementació o bé en veure els resultats de l'enquesta feta a la memòria i de la que no es disposava abans de començar el desenvolupament de l'app. Com es pot veure a la figura 1.1, el desenvolupament d'un software és una feina circular que sempre es va redissenyant, implementant, posant-se a

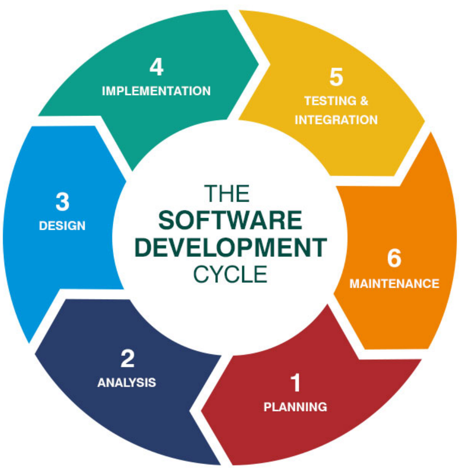


Figura 1.1: Cicle del desenvolupament del software. Font: `http://online.husson.edu/software-development-cycle/`

prova, i tornant a començar. És per això doncs, que al llarg d'aquesta memòria es podrà veure com a mesura que evoluciona el projecte, i posant a prova algunes de les funcions inicials, aquestes aniran variant de costat amb l'evolució del projecte.

La memòria està estructurada de la manera més cronològica possible des de la idea inicial fins al prototip. Just després d'aquesta introducció hi ha el capítol relacionat amb l'enquesta anteriorment mencionada que prova d'englobar el màxim de visions i maneres de fer del protocol d'enregistrament per tal de fer una app compatible amb totes elles. Seguidament, Al Capítol 3 *Disseny funcional*, es podrà veure quins passos s'han seguit per a dissenyar el funcionament de l'aplicació. En aquest capítol, encara quasi no es parlarà de codi en si, simplement es voldrà establir un funcionament de l'aplicació mitjançant un mockUp i preparar el terreny perquè la implementació sigui fluida. No serà fins al següent capítol (Implementació) on s'implementarà tot allò que s'ha dissenyat, ara sí, mitjançant el codi Swift i sobre l'iPad.

Tot i no haver-se pogut dur a terme encara, el capítol 5 *Passos a seguir després del prototip* intenta resumir els passos claus que caldrà dur a terme des que es presenta el prototip fins a la sortida al mercat de l'aplicació. Finalment i ja per acabar, hi haurà les conclusions generals del treball.

Tots aquells conceptes propis del llenguatge de programació i d'altres no molt comuns, aniran comentats amb una nota al peu de pàgina per tal de no fer feixuga la lectura a l'haver d'explicar-los. Generalment, l'explicació conduirà a un web extern (Ja sigui Wikipedia, el web del desenvolupador, recursos de Swift, etc...). La versió digital d'aquesta memòria que es pot trobar al web `www.cassorla.cat/memoriaTFG` té incrustats enllaços que redirigeixen directament a les cites, siguin externes o internes entre capítols o annexos.

CAPÍTOL 2

TREBALL DE CAMP

2.1 Objectius

Com s'ha comentat 1r capítol d'aquest treball. La informació sobre el protocol d'enregistrament és escassa, per no dir gairebé nul·la. Els apunts del Pere Casulleras són l'única informació que s'ha pogut trobar i si l'objectiu és fer una eina útil per a la comunitat sonològica (a escala global), aquesta no pot estar basada només en els apunts d'una assignatura relativament petita d'una sola universitat. És per això que havent arribat a la conclusió que, com diria la dita castellana, *cada maestrillo tiene su librillo* s'ha decidit recopilar tots aquests *librillos* per a fer una aplicació que pugui arribar al màxim de visions possibles del protocol d'enregistrament.

L'objectiu principal és ratificar que fora d'aquí també es fa servir aquest tipus de nomenclatura i en cas afirmatiu, quines són les seves variants. Veient reportatges d'enregistraments de fora, es pot veure en més d'una ocasió el tonmeister prenent apunts sobre la partitura, així com en webs d'empreses d'enregistrament de música clàssica[1] es veuen imatges on tenen partitures *guixades* amb uns símbols semblants als que es fan

[1] http://www.classicalmedia.co.uk/post-production/music-editing/

servir aquí. És més, en Pere Casulleras, qui ha ensenyat la metodologia dels protocols d'enregistraments a la fornada de productors catalans, va aprendre-ho a Suïssa. Per tant, és molt probable que sí, fora d'aquí també es faci servir aquest tipus de protocol, però el que no se sap és de quina manera s'utilitza i quines variants té.

Un objectiu secundari, però que no deixa de ser important, és veure fins a quin punt els professionals del sector veuen amb bons ulls l'aplicació que s'està desenvolupant i també a la vegada, començar a preparar el terreny i donar a conèixer l'app abans que surti al mercat.

2.2 Abast

Si l'objectiu és arribar a comprendre el màxim de visions possibles, evidentment l'abast de l'enquesta serà el màxim al qual pugui arribar. És per això que l'enquesta està plantejada en anglès i així pot tenir un abast internacional.

Per una banda, s'han utilitzat els contactes personals de productors i estudiants, així com d'altres propers als primers esmentats per a fer-ne difusió. Per altra banda, en el marc internacional s'han recopilat una sèrie de contactes de *lliure accés* que s'han trobat per la xarxa. Concretament contactes de professors d'universitats qui tenen un correu de contacte penjat públicament al web de la universitat. S'han pogut aconseguir uns vint contactes de cinc universitats diferents: Surrey[2] (Anglaterra), Detmold[3] (Alemanya), MDW[4] (Àustria), McGill[5] (Canadà) i RDAM[6] (Dinamarca). En altres universitats o bé empreses d'enregistrament de música clàssica, hi ha un formulari de contacte a Internet on també s'hi podrà enviar l'enquesta.

Als contactes personals com poden ser les llistes de sonologiaEsmuc o bé gent d'aquí Catalunya, s'ha enviat un correu *genèric* explicant de què va el treball, demanant

[2]`https://www.surrey.ac.uk`
[3]`https://www.eti.hfm-detmold.de`
[4]`https://www.mdw.ac.at/`
[5]`https://www.mcgill.ca/`
[6]`https://english.dkdm.dk/`

que contestin l'enquesta i que la facin extensible a qui creguin que li pugui interessar. Als contactes internacionals en canvi, sel's hi ha enviat un correu personalitzat, presentant l'autor, presentant el treball i sobretot també, demanant que a banda de contestar-la, passin l'enquesta a tota la gent que puguin, tant siguin alumnes o companys de feina dels qui no s'ha pogut aconseguir el contacte.

Per acabar, també s'ha fet difusió a fòrums especialitzats com Gearslutz[7], Sound on Sound[8], AES[9], etc...

En tot això, s'ha establert un termini per poder tenir les respostes en un cert marge de temps i, en el cas de no rebre resposta, enviar un segon correu recordatori quan s'acosti la data.

2.3 Continguts

El contingut de l'enquesta s'ha elaborat de tal manera que no prengui a l'enquestat més de 5 minuts. Per a fer això i a la vegada poder treure tota la informació necessària, s'han preparat unes preguntes molt concretes i sempre que s'ha pogut, amb opcions preestablertes a la resposta. Aquestes preguntes estan dividides en 5 grans blocs:

2.3.1 Classical recording protocol (presentació):

Aquest bloc no conté cap pregunta, simplement posa en context l'enquestat sobre per què s'està fent aquesta enquesta i sobre el protocol d'enregistrament en si. Part d'aquesta informació també està al correu electrònic que s'ha enviat als enquestats però com la intenció és que l'enquesta circuli, s'ha cregut convenient posar la presentació per si a algú li arriba a les mans l'enquesta sense saber de què va la cosa.

[7]https://www.gearslutz.com/
[8]http://www.soundonsound.com/
[9]http://www.aes.org/

2.3.2 About you:

En aquest bloc s'ha volgut extreure una mica d'informació sobre l'enquestat. Hi ha tres preguntes voluntàries com són el nom, el correu electrònic i el web. Aquestes preguntes són voluntàries, ja que realment no és necessari per al treball saber qui ha respost cada cosa. El que sí que és interessant i s'ha marcat com a obligatori és dir des de quin país s'està responent l'enquesta, quina carrera i a quina universitat s'ha estudiat i també si actualment aquella persona es dedica a la música.

Sabent els llocs i/o universitats de procedència, es podran agrupar els resultats i poder veure la manera de fer els protocols d'enregistrament a cada lloc. Sobretot si són molts els resultats obtinguts, pot ser una eina molt útil. Per altra banda, sabent si l'enquestat està treballant o no actualment a la indústria de la música, es podrà saber si les seves respostes són més o menys *actualitzades* respecte al que actualment s'està duent a terme.

2.3.3 About the classical recording protocol:

Primer de tot es pregunta si ha sentit a parlar sobre el tema i en cas afirmatiu si utilitza el protocol habitualment, poc o gens. En les possibles respostes, a banda del "Yes"o "No", s'ha afegit l'opció "Yes, but with another name", ja que és molt probable que no tothom utilitzi aquest nom. En el cas que l'enquestat marqui aquesta opció, hi ha una pregunta a continuació on es demana quin és l'altre/s nom/s que coneix.

Les preguntes que hi ha a continuació són específiques sobre com l'enquestat fa els seus protocols d'enregistrament. Primer es pregunta si utilitza uns símbols comuns *apresos* o bé uns de seus propis. Després, sobre si marca l'inici i final de les preses a la partitura (donant l'opció de respondre "Only the start point"o "Only the end point"), sobre si marquen les preses bones, dolentes, totes dues o "altres", i també si sempre comencen els enregistraments per la presa número 1 (o en cas contrari quin és el criteri de numeració).

Les respostes obtingudes d'aquest apartat seran de gran utilitat per poder adaptar el disseny de l'app perquè a tothom li sigui el més familiar possible. Per exemple: si es dóna el cas que de normal començaria els enregistraments per la presa u, però a l'enquesta es veu reflectit que molts dels enquestats no ho fan, és relativament senzill posar una opció quan es carrega un fitxer que permeti establir la presa d'inici d'aquella peça.

2.3.4 About the app:

Si en el bloc anterior es pregunta com es fan actualment els protocols d'enregistrament, en aquest es demana a l'enquestat si creu que, tal com fa actualment els protocols, podria fer-ho mitjançant una aplicació. En aquesta pregunta s'han donat quatre opcions diverses, ja que no és qüestió de sí o no. Potser ara mateix no ho veu, però quan vegi l'app sí que ho veu factible.

En aquest bloc també s'ha preguntat sobre què s'esperaria d'una aplicació com aquesta. És una pregunta molt oberta, però d'aquí se'n podran treure moltes idees de cara a futures actualitzacions o també endreçar les idees inicials. L'última pregunta d'aquest bloc és si volen rebre informació quan surti l'app, fent així una llista de *possibles compradors* de l'aplicació.

2.3.5 References:

Finalment, aprofitant la difusió, es demana als enquestats si saben de l'existència d'algun llibre o article relacionat amb el protocol d'enregistrament. Fins i tot, alguns apunts de classe com els esmentats de en Pere Casulleras però d'alguna altra universitat. Es deixa un espai per escriure tot i que també hi ha la possibilitat d'enviar-ho per correu electrònic.

Ja al final de l'enquesta, quan es donen les gràcies per participar-hi, es demana un últim favor. Si volen ajudar una mica més encara, poden enviar una fotografia d'un protocol d'enregistrament seu i així es podrà estudiar més a fons. S'ha decidit posar-ho al final i "fora de l'enquesta", ja que s'entén que és una cosa més pesada que no pas respondre les preguntes.

2.4 Resultats

A dia 26 d'abril s'han recopilat els resultats de l'enquesta que, tot i haver estat oberta durant poc més d'un mes, ha aconseguit reunir més de 70 respostes. En aquesta secció es pot veure un resum de les respostes obtingudes a l'enquesta, tot i així, si es volen consultar els resultats i respostes íntegres, es poden trobar a la pàgina 61 de l'Annex.

S'han rebut respostes de 14 països diferents on es poden destacar Anglaterra, Catalunya, els Estats Units i Àustria com els que més. Les universitats de procedència dels enquestats són molt diverses però s'han rebut forces respostes de l'ESMUC (Catalunya), Surrey University (Anglaterra), la MDW (Àustria) i la McGill University (Canadà), universitats a les quals se'ls havia enviat una petició de difusió de l'enquesta. Els estudis realitzats per les persones que han contestat l'enquesta són molt diversos, l'opció *Sonologia* només l'han respost els enquestats de l'ESMUC (15,3%) mentre que el 36,1% han realitzat o estan realitzant estudis de Tonmeister i un 22,2% han realitzat altres estudis relacionats amb el so. El 26,4% restant són altres estudis, alguns més relacionats amb la música que d'altres, on es pot veure que varis provenen de la interpretació instrumental. De tots els enquestats, el 94,4% està en actiu professionalment.

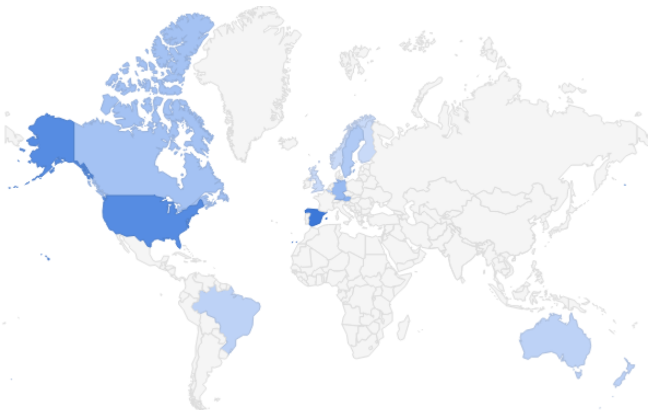


Figura 2.1: Distribució al mapa dels països de procedència dels enquestats

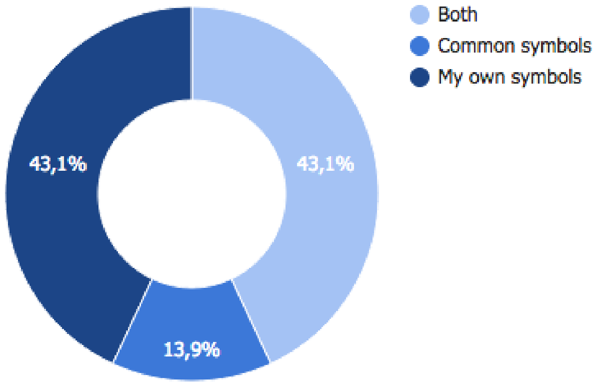


Figura 2.2: Percentatges de les respostes a la pregunta *Utilitza símbols propis o d'apresos?*

Al preguntar sobre si es coneix el protocol d'enregistrament, el 68,1% diu que n'ha sentit a parlar, sigui amb aquest nom o bé amb un altre, mentre que el 31,9% diu que no n'ha sentit a parlar. Val a dir que molts dels que han contestat que *no n'han sentit a parlar*, a la següent pregunta han respost que sí que l'utilitzen. Segurament la pregunta era massa ambigua i algú pot no haver sentit a parlar del nom *Protocol d'enregistrament* però sí del concepte, que és el que s'intentava preguntar. Els percentatges resultants d'aquesta pregunta doncs, no acabarien de ser del tot fiables. Tant dels que han respost que sí, i com s'acaba de comentar, alguns que han respost que no, el 68,1% dels enquestats diu que l'utilitza regular o esporàdicament mentre que el 31,9% restant no.

Sobre la manera d'utilitzar-lo hi ha una clara victòria dels símbols propis vers als *apresos*, sigui utilitzant-ne només de propis o combinats amb els *apresos*. Tal com es pot veure a la Figura 2.2, només el 13,9% utilitza únicament símbols *apresos* i no n'utilitza de propis. Dins d'aquests símbols, un 65,3% dels enquestats marquen tant l'inici com el final de presa a la partitura mentre que un 12,5% només marca l'inici i un 20,8% ha respost que no ho marca. També queda bastant clar que gairebé tots els enquestats marquen tant preses bones com dolentes, ja que el 62,5% han respost això mateix, però a aquest percentatge se li hauria de sumar el 22,2% que ha respost *altres*, ja que simplement han especificat matisos com el tipus d'error, si una presa és molt més bona que una altra, etc... Només el 13,9% ha respost que marca les preses bones i tan sols l'1,4% ha dit que marca només les dolentes.

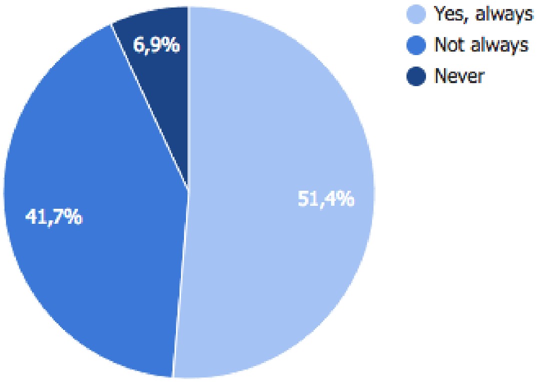


Figura 2.3: Percentatges de les respostes a la pregunta *Normalment comença els enregistraments des de la presa 1?*

A la pregunta *Normalment comença els enregistraments des de la presa 1?* hi ha hagut un clar empat entre les opcions *Sí* i la suma de *No sempre* i *mai*, tal com es pot apreciar a la Figura 2.3. Aquesta pregunta anava acompanyada d'una altra opcional que preguntava quin era el criteri en cas de no fer-ho (o no sempre). Aquí hi ha hagut respostes de tots els gustos i colors, però també algunes que posaven de manifest que la pregunta no era prou clara, ja que no s'especificava si el *començar* es referia a cada vegada que es prem el *rec*, quan es comença una nova peça, una nova sessió o un nou projecte.

Entrant ja a l'apartat que pregunta sobre l'aplicació, una àmplia majoria creu que podria prendre notes tal com les pren ara però mitjançant l'iPad, ja sigui amb un sí incondicional (45,8%), un sí, encara que a hores d'ara costi d'imaginar (29,2%) o no s'ho imaginen però estarien disposats a provar-ho (18,1%). Només un 6,9% ha respost que no ho veu possible. La pregunta *Que espera d'una aplicació per a fer protocols d'enregistrament?* era la més oberta de totes i on la majoria d'enquestats s'hi ha estat més estona responent. Hi ha hagut respostes molt diverses on per una banda demanen coses molt bàsiques que ja estaven contemplades a la idea inicial del projecte i per altra banda donen moltes noves idees de cara a futures millores. Tot i així, les respostes es poden resumir dient que sobretot esperen de l'aplicació rapidesa a l'hora de prendre notes i que sigui igual o més pràctic que prendre-les en paper. A partir de la pàgina 70 de l'annex es poden llegir totes les respostes a aquesta pregunta.

La pregunta opcional que preguntava si es té alguna referència parlant sobre el protocol d'enregistrament només l'han respost el 5,5% dels enquestats i tot i així, no hi ha hagut massa informació. S'ha parlat del capítol 20 d'un llibre recentment publicat per Focal Press (King, 2017) on es pot trobar una guia bàsica sobre el protocol i també s'han rebut un parell d'enllaços al web d'un software molt antic que intentava fer quelcom semblant al protocol d'enregistrament. Finalment, el 86,1% dels enquestats té interès en què se'l mantingui informat un cop l'aplicació surti al mercat.

2.5 Conclusions de l'enquesta

Els resultats obtinguts a l'enquesta són molt positius per al projecte, ja que posen en evidència la necessitat que té el sector de poder utilitzar una eina com la que s'està desenvolupant. Si més no, s'ha pogut comprovar que l'interès perquè hi sigui hi és. Tant en moltes respostes a la mateixa enquesta, en els fòrums on s'ha fet difusió i fins i tot en respostes per correu dels contactes a qui se l'ha enviat, s'han rebut mostres de suport i fins i tot oferiments d'ajuda per tirar endavant el projecte. Només cal mirar els percentatges de les últimes preguntes (Figura 2.4) on es veu reflectit aquest interès i per tant, els objectius principals estan assolits.

L'apartat que més ha fet redissenyar el projecte ha estat el que preguntava la manera de fer els protocols d'enregistrament dels enquestats. Tal com s'ha pogut veure a la Figura 2.2, el fet que molts dels enquestats utilitzi símbols propis obliga a donar

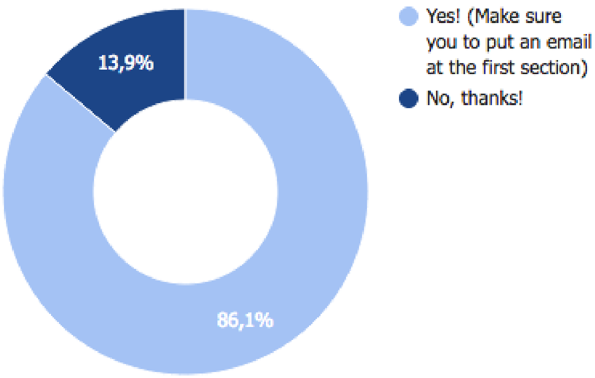


Figura 2.4: Percentatges de les respostes a la pregunta *Vol estar informat quan l'aplicació surti al mercat?*

l'opció de poder introduir-los d'alguna o altra manera (sigui dibuixant, o entrant un arxiu d'imatge), per contra tota aquesta gent no utilitzaria l'aplicació... També s'ha vist que no tothom comença els enregistraments a la presa 1 (Figura 2.3) així que l'opció d'entrar el número d'inici de presa manualment s'haurà d'afegir al projecte. Per acabar, la idea inicial obligava a posar el final de presa abans d'avançar a la presa següent, però com no tothom el posa normalment, aquesta opció haurà de passar a ser només un avís i no una obligació. En resum, hi ha moltíssimes maneres de fer diferents que obligaran a dissenyar una aplicació totalment personalitzable.

Per altra banda, ha estat molt interessant veure que les principals preocupacions dels enquestats eren coses que ja estaven contemplades a la idea inicial del projecte. Per exemple, molts demanaven que les anotacions es poguessin canviar de color i de mida, que es pogués escriure en llapis (digital) directament a la partitura, que el canvi de pàgina fos ràpid, prendre notes amb símbols preestablerts i tot de qüestions amb les quals ja es comptava. Fins i tot, enquestats que han respost que no creuen que sigui possible dur a terme els protocols d'enregistrament amb l'iPad, els arguments que han donat són precisament aquest tipus de demandes i per tant, fins i tot es podria arribar a aquesta gent.

A la pregunta oberta del final s'han rebut molts suggeriments que, encara que no serien necessaris d'implementar per al moment de llançar l'aplicació al mercat, són molt bones idees per anar afegint una vegada ja estigui en funcionament i per tant, serviran per a fer noves versions i poder-la anar actualitzant.

CAPÍTOL 3

Disseny funcional

3.1 Esquema i MockUp

El primer pas a realitzar un cop la idea era clara ha estat esquematitzar-la i deixar-la per escrit. Per a fer-ho s'ha utilitzat el recurs prezi, una eina que permet fer esquemes dins d'altres esquemes i així poder tenir una visió global de tot el projecte. Es pot veure el resultat a l'enllaç del peu de pàgina o bé a la pàgina 78 de l'Annex. L'esquema divideix per primera vegada l'aplicació en tres grans seccions segons els diferents passos d'una producció i també en una secció *home*:

HOME: La secció *home* serà accessible des de qualsevol de les altres tres seccions i permetrà poder organitzar els fitxers per carpetes com a l'usuari més li convingui, descarregar-les dels diferents serveis d'emmagatzematge o bé eliminar-les.

PRE: En l'esquema original fet amb prezi[1], la secció *home* estava inclosa dins de la secció "PRE", fent el mockUp però, es va creure més convenient posar-la per separat, ja que era més còmode i intuïtiu. A la secció "PRE"es podrà preparar la partitura de

[1] `http://prezi.com/ivxtx6vkwono/?utm_campaign=share&utm_medium=copy`

cara a l'enregistrament. Ja sigui fent anotacions *a ma*, posant *punts de llibre* o també separant els sistemes d'una mateixa pàgina, funció que serà de gran ajuda quan es vulgui treballar amb les automatitzacions de l'app.

REC: En aquesta secció és on es faran totes les anotacions durant l'enregistrament. Aquestes anotacions poden ser preestablertes (com per exemple els inicis/finals de *take*, +/-/+-/ok, etc...), anotacions *a llapis*, notes (escrites) per presa o globals de tota la peça, etc... Totes aquestes notes estaran basades en un *take* en concret que es podrà veure a la part superior de la pantalla acompanyat de dos botons per canviar de presa. És aquí on es podran començar a notar els avantatges dels processos automatitzats. Per exemple: l'aplicació podrà detectar si algun sistema encara té algun error sense solucionar o si pel contrari, la peça ja té com a mínim una versió bona de tots i cadascun dels sistemes/pàgines.

POST: Per acabar, un cop s'han pres totes les anotacions pertinents, en aquesta secció es podrà treballar amb elles de diverses maneres. Les dues seccions anteriors són (exceptuant les automatitzacions esmentades) una versió digital de l'actual protocol d'enregistrament. Aquesta secció "POST"en canvi, és la secció on hi haurà més innovació, ja que s'hi podran fer moltes coses que fins ara eren pesades i/o lentes de fer amb una partitura a paper. Per exemple es podrà veure quines preses hi ha a un punt concret, filtrar-les segons la valoració o fins i tot l'app podrà fer una proposta de pla de muntatge.

Després d'esquematitzar les idees, tenir clara la funcionalitat de l'aplicació i haver-la dividit en les seves tres seccions, s'ha procedit a fer un mockUp d'aquesta. Aquest mockUp s'ha realitzat amb l'entorn balsamiq[2], una eina que permet fer un dibuix de les diferents seccions de l'aplicació i posar enllaços d'un dibuix a un altre per a poder veure un primer funcionament. Sense haver escrit ni una sola línia de codi, ja es pot veure l'esborrany de les funcions que farà l'app i en aquest punt s'ha presentat a diversos

[2]`https://balsamiq.com/`

professors i companys perquè en valorin la seva funcionalitat. Com es pot veure a la figura 3.1, el disseny és molt simple però és suficient per fer-se'n una idea. A la pàgina 81 de l'Annex s'hi pot trobar el mockUp sencer.

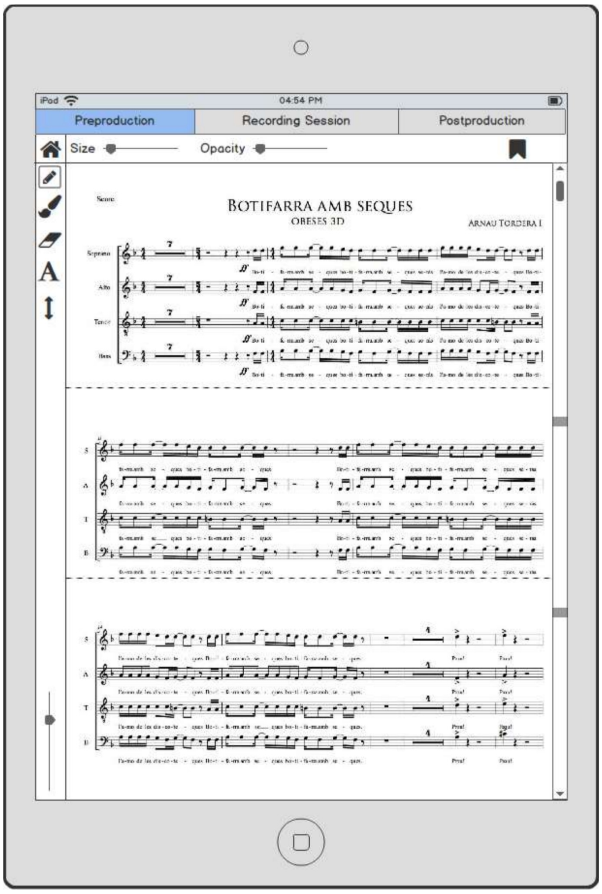



Figura 3.1: Mostra de l'esborrany de l'aplicació realitzat amb balsamiq

Un cop acabat l'esborrany i havent-ho comentat amb d'altra gent, l'esquema inicial va patir alguns canvis en aspectes que, després de veure aquesta petita demostració del funcionament que ofereix balsamiq, es va creure convenient modificar. També aquí es van afegir funcionalitats més concretes que no pas a l'esquema inicial com per exemple, el fet de poder canviar l'opacitat de les anotacions, veure en quin estat està la peça (si enregistrada, acabada o per començar), etc.

La figura 3.1 correspon a la secció PRE però com es pot observar, gran part de la pantalla està ocupada per la partitura. Aquesta regió serà comuna a les tres seccions de l'aplicació i de fet, l'únic element que canviarà serà la barra de l'esquerra i la barra superior (sota els botons). En aquestes barres hi apareixeran o s'amagaran les opcions pertinents de cada secció. Per canviar de secció s'haurà de clicar als botons de la part superior i per accedir al *home* a la icona situada on s'ajunten les dues barres.

L'esquema en prezi i les primeres funcionalitats aplicades al mockUp han estat el punt de partida de la programació i és gràcies a això que s'ha pogut seguir un ordre molt clar per programar. Com s'explicarà a la següent secció, el fet de tenir la idea clara abans de començar a escriure codi ha estat un factor clau d'aquest projecte, ja que qualsevol pas endavant que es feia es procurava fer de tal manera que facilités la feina als passos següents.

3.2 Metodologia de programació

La metodologia de programació ha estat basada a anar implementant de gran a petit, de qüestions més bàsiques a més detallades però tot i així, sempre tenint al cap i molt en compte l'objectiu final al qual es vol arribar. S'ha començat a programar des d'un pla molt global i cada vegada s'ha anat acotant més tal com es detalla a continuació:

El primer pas ha estat passar el disseny del mockup a l'Xcode[3] (l'entorn de programació d'Apple) i poder navegar per les seccions. Un cop assolit això el punt de partida ha de ser poder llegir les partitures dels usuaris. Per una banda s'ha de decidir el format en el qual es visualitzaran les partitures i per altra banda decidir a través de quin mitjà l'usuari les passarà del seu ordinador/correu-electrònic/núvol/qualsevol-altre-lloc a l'aplicació. No es descarta que alguna actualització estigui oberta a altres formats, però com a punt de partida s'ha cregut oportú utilitzar el PDF, ja que des de qualsevol altre format és relativament fàcil passar a PDF, en canvi a l'invers és complicat o directament no és possible.

Per obtenir els fitxers, segurament l'opció més fàcil seria que l'usuari connectés el seu dispositiu a l'ordinador mitjançant el cable USB i que introduís els fitxers a l'iPad a

[3] `https://developer.apple.com/xcode/`

través de l'iTunes. És l'opció més fàcil a nivell de programació, però posant-se al nivell de l'usuari és una opció no massa atractiva. La millor opció és poder descarregar-se del núvol les partitures mitjançant serveis de 3rs com Dropbox entre molts d'altres. El primer pas doncs, haurà de ser poder descarregar-se aquestes partitures, mostrar-les per la pantalla de l'iPad i poder-hi accedir des de les diferents seccions.

Com tots els passos que s'han anat fent sempre miraven a l'objectiu final així que el següent pas a realitzar ha estat aconseguir separar els sistemes d'una pàgina. Per poder ser acurats a l'hora de desar una anotació i després poder treballar amb ella en relació a la partitura aquest pas era molt necessari, ja que d'altra manera no es podria saber si una anotació va abans o després en la música. Quan es prem la pantalla per introduir una nova nota, el mateix llenguatge d'Apple podrà dir en quina x i quina y exactament s'ha premut. Posant per cas que la partitura va a sistema per pàgina, només sabent quina pàgina i quina x té el punt on està l'anotació, es podrà saber si una nota va abans o després en la música, ja que es llegeixen els pentagrames d'esquerra a dreta. Ara bé, si una partitura té més d'un sistema per pàgina (com la de la Figura 3.2) s'haurà de dividir aquesta en dues, tres o tantes parts com sistemes tingui per poder determinar l'ordre en el temps de les notes preses. En cas de no fer-ho així, en aquest exemple el programa detectaria que la nota número 3 va abans que la 2.

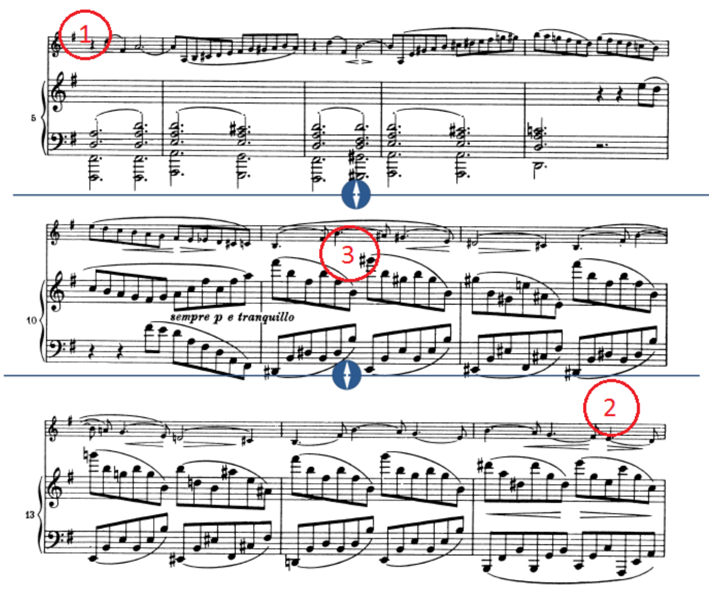



Figura 3.2: Exemple de partitura amb més d'un sistema per pàgina

Un cop es té el punt i el sistema premut, s'han d'emmagatzemar aquestes dades i totes les relacionades amb una anotació en concret. Al capítol 4 es parlà exactament de quines dades es desen i perquè però bàsicament, a banda de la posició, es necessitarà saber el sistema on es troba, la pàgina, de quin document és i de quin tipus de nota es tracta. La idea inicial era de poder escriure diferents tipus de notes a la partitura en funció del nombre de clics efectuats. Si se'n fa un, una nota, si se'n fan dos una altra, i així fins a cinc seguits. És per aquest motiu que el següent pas ha estat poder escriure diferents tipus de notes en funció dels clics que es fan i també poder canviar de nota o eliminar-la un cop ja ha estat inserida. Totes aquestes dades han de poder ser accessibles quan es torna a carregar el mateix document o bé s'ha reiniciat l'aplicació.

Aquestes notes però, ara mateix estan escrites en un sol *take*, així que per poder fer un protocol d'enregistrament encara falta poder canviar de presa i que les anotacions que es vagin introduint estiguin relacionades a aquest *take* en concret. Un cop aconseguit, l'aplicació és totalment funcional i s'hi pot fer un protocol més o menys com es faria en paper. Però no només això, sinó que totes aquelles anotacions que es van prenent ja tenen emmagatzemada la informació necessària per després poder treballar amb elles i fer totes les funcions extres que es vulgui. Aquestes funcions extres s'aniran introduint a partir d'aquest punt en funció de la prioritat que se li vulgui donar fins a assolir el punt en què l'aplicació pugui sortir al mercat.

Un altre aspecte a tenir en compte de la metodologia de programació és que qui està utilitzant l'aplicació, pot no saber com funciona alguna cosa i per tant utilitzar-la de manera errònia i fer "petar"el codi. Per evitar-ho és molt important imaginar totes les possibilitats d'interacció que té l'usuari amb cadascuna de les eines que es van incorporant per tal d'evitar errors abans que es produeixin.

3.3 Llibreries i APIs

Hi ha funcionalitats de l'aplicació que s'està desenvolupant que no serà necessari de programar de nou, ja que moltes altres aplicacions fan aquella funció i no hi ha res de nou que l'aplicació en qüestió necessiti aportar. Per exemple el fet de veure un arxiu PDF per pantalla. Programar de zero un visor de PDF prou eficient i ràpid és una feina molt feixuga i difícil que es pot estalviar utilitzant una llibreria. Per altra banda, quan es vol interactuar amb algun servidor extern com pot ser Dropbox, es necessita utilitzar una API (Application Programming Interface) que proporcionarà el mateix servei.

3.3.1 Llibreries

Una llibreria és un conjunt de subprogrames que contenen el codi necessari per dur a terme alguna funció. En aquest cas, s'ha instal·lat una llibreria que llegeix arxius PDF i, des del codi propi, només s'ha de cridar una funció d'aquesta llibreria per tal de veure el PDF en pantalla.

De llibreries que llegeixin arxius PDF hi ha moltíssimes, algunes són de pagament i d'altres de codi obert. En aquest cas, com a primera opció s'ha intentat utilitzar una llibreria de codi obert. Moltes llibreries utilitzen una funció del mateix llenguatge de programació d'Apple que es diu WebView[4]. Aquesta funció és una mena de navegador web on es pot veure el PDF i a més, té incorporades diverses utilitats com per exemple el zoom fent doble clic, passar de pàgina d'una manera molt fluïda, etc. Si només es volgués visualitzar el PDF, aquest tipus de visor seria suficient però és massa tancat i no es pot treballar bé amb ell per tal d'implementar tota la resta de funcions. D'altres llibreries que s'han provat passaven la pàgina de tal manera que depenent de la velocitat amb què es feia passar, passava una o fins a més de deu pàgines. Quan s'està enregistrant, es necessita que les coses siguin simples i fàcils d'utilitzar així que haver de calcular la velocitat amb la qual es passarà la pàgina no era una bona opció. Finalment, tot i tenir

[4] `https://developer.apple.com/reference/uikit/uiwebview`

una dificultat afegida a l'hora d'implementar-la, s'ha trobat la llibreria adequada que permet treballar bé amb ella, té un canvi de pàgina simple i és molt eficient.

3.3.2 APIs i SDK

Així com les llibreries utilitzen recursos propis del llenguatge de programació per a fer alguna funció concreta, les APIs per la seva banda es comuniquen amb un servidor extern per a intercanviar dades. Existeixen APIs de tota mena, des de Xarxes Socials fins a bases de dades en línia molt complexes. En el cas de l'aplicació que s'està desenvolupant, s'hauran d'utilitzar APIs d'emmagatzematge com són Dropbox[5], OneDrive[6], GoogleDrive[7], Mega[8], etc... Cadascuna d'aquestes s'implementarà d'una manera diferent, però generalment són de lliure accés i estan molt ben documentades a la secció *developers* que gairebé tots aquests serveis disposen al seu web. Tot i així, per a la presentació del prototip només s'implementarà una per fer la prova. L'opció triada és Dropbox, ja que és de les més comunes i l'API és senzilla d'utilitzar.

Per altra banda, hi ha els SDK (Software Development Kit) que entre molts altres usos, poden contenir eines per comunicar aparells externs amb l'aplicació. Per exemple s'utilitzen per poder utilitzar un bolígraf digital com wacom[9] o adonit[10], eina molt útil i demanada per molts usuaris a l'enquesta.

Un cop triades les llibreries, APIs o també alguns SDks, encara abans de començar a programar s'han d'instal·lar al projecte mitjançant cocoapods[11], un petit programari que s'instal·la al terminal de l'ordinador i permet incrustar les llibreries de manera molt fàcil i ràpida als projectes d'Xcode.

[5] https://www.dropbox.com/developers
[6] https://dev.onedrive.com/
[7] https://developers.google.com/drive/
[8] https://docs.mega.nz/sdk/doc/api.html
[9] https://developer.wacom.com/
[10] http://www.adonit.net/developers/
[11] https://cocoapods.org/

CAPÍTOL 4

IMPLEMENTACIÓ

4.1 Llenguatge i entorn programació

Per poder començar a programar primer s'ha de conèixer el llenguatge de programació amb el qual es farà. Les aplicacions per iOS es poden programar bàsicament amb dos tipus de llenguatge: En Objective-C o bé en Swift. L'Objective-C va ser el primer llenguatge que permetia desenvolupar aplicacions per iOS. Com explica la introducció del llibre *Objective-C programming* (Hillegas and Ward, 2013), és un llenguatge orientat a objectes que afegeix l'estil "Smalltalk"[1] al llenguatge C. L'any 2014 però, Apple va presentar el que avui en dia és el llenguatge més utilitzat en el desenvolupament d'aplicaccions iOS, OSX i AppleTV: el Swift.

El llenguatge Swift va ser creat inicialment com a llenguatge propi d'Apple. Estava inspirat en l'Objective-C però amb l'objectiu de fer més fàcil la programació eliminant aspectes heretats de C com els punts i comes al final de línia, molts parèntesis obligatoris en algunes funcions o el fet d'haver d'assignar un tipus de variable al declarar-la entre moltes d'altres novetats. L'any 2015, tot just un any després del seu llançament, el

[1] `https://en.wikipedia.org/wiki/Smalltalk`

llenguatge va passar a ser *open source* obrint-se així a altres plataformes com Linux. Segons indica el web oficial[2] els avantatges que té Swift són: la seguretat (sent un llenguatge estricte però a la llarga molt segur), la rapidesa (gràcies a la simplificació del llenguatge) i l'expressió (tenint una sintaxi clara i entenedora).

En el cas d'Apple i concretament el desenvolupament d'aplicacións mòbils iOS, l'entorn de programació per a fer-ho és l'Xcode[3]. L'Xcode, a banda de ser un gestor de fitxers (tant fonts, com imatges o altres arxius que necessiti l'aplicació), tenir una consola integrada, i les eines habituals de les quals disposaria qualsevol altre IDE[4], té una eina molt interessant i útil per poder relacionar la part gràfica de l'aplicació amb el codi. Si bé es podria escriure una aplicació sencera només utilitzant codi i sense aquesta eina, hi ha molts casos que és de gran ajuda i estalvia molt de temps. L'eina per exemple, permet fer una storyBoard completa de l'app, insertar botons, textos, imatges... entre altres funcionalitats. En el cas pràctic que s'està tractant, ha estat molt útil sobretot pel que fa al canvi de seccions i diferents "finestres"de l'aplicació.

Com es pot veure a la Figura 4.1, hi ha moltes opcions preestablertes de disseny

[2] `www.swift.org`
[3] `https://developer.apple.com/xcode/`
[4] Integrated development environment

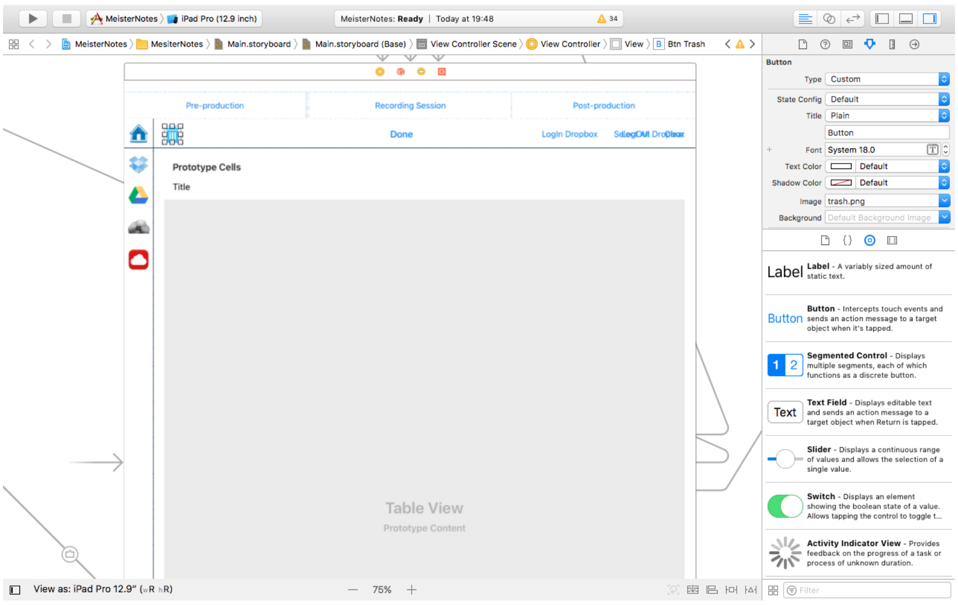


Figura 4.1: Captura de l'eina gràfica d'Xcode-8

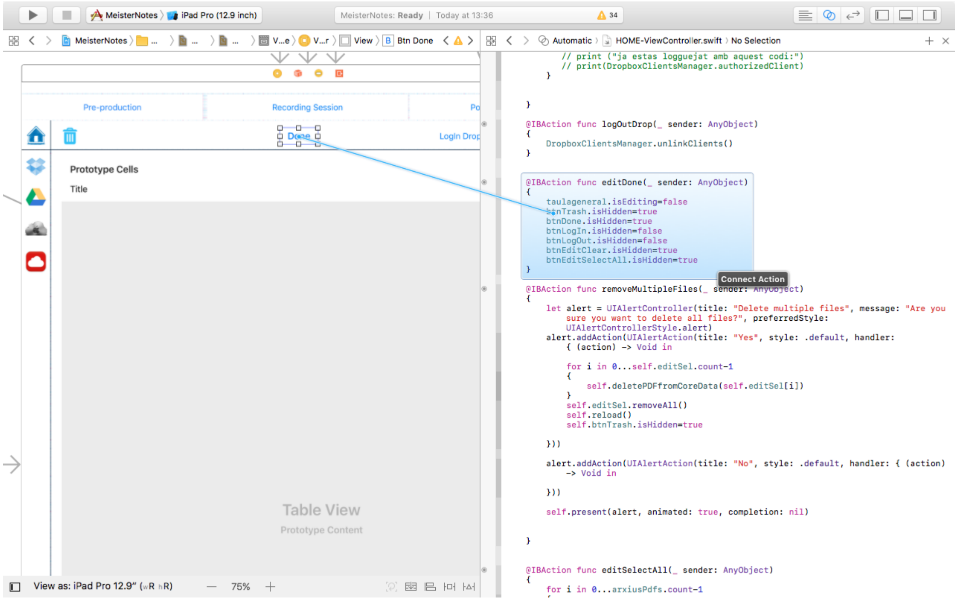


Figura 4.2: Captura de la interacció gràfics-codi

(en l'estil d'Apple) que permeten fer un prototip del que se'n parlarà més detalladament a la secció 4.4 *Prototipatge*. El que és molt útil també és l'opció de posar la pantalla partida (Figura 4.2) on es pot veure el codi i també la part gràfica. En aquest mode, es pot arrossegar un botó que s'ha posat mitjançant l'eina gràfica, al codi. D'aquesta manera quan es premi aquell botó, l'aplicació correrà el tros de codi que s'hagi establert per aquella acció.

4.2 Flux de dades

Un aspecte important a tenir en compte un cop es comença a programar és quina gestió de la memòria RAM es farà i què/com/quan es desarà a la memòria interna del dispositiu. La memòria RAM[5] és una memòria d'accés molt ràpid i que per tant serà interessant d'utilitzar mentre l'aplicació estigui activa, per contra té molt poca capacitat i cada vegada que es tanca l'aplicació s'esborrarà el seu contingut. Per altra banda hi ha la memòria interna de l'iPad, que en aquest cas es tracta d'una memòria SSD (no molt més lenta que la RAM) això sí, amb molta més capacitat. La part bona de la memòria interna (a banda de la capacitat) és que no s'esborra al tancar l'app però programaticament és

[5] https://en.wikipedia.org/wiki/Random-access_memory

més complicat d'accedir-hi. Veient que tots dos tipus de memòria tenen les seves parts bones i les seves dolentes, s'haurà de dissenyar un bon flux de dades entre una i altra memòria. Les dades que es necessitaran desar són les següents:

4.2.1 Arxius

El primer que l'usuari haurà de fer un cop s'instal·li l'app, abans de començar a treballar, serà descarregar-se alguna partitura en format PDF. Aquests arxius es descarregaran directament a la memòria interna del dispositiu. Per començar anirà bé tenir-ho a la memòria interna i no obrir-ho mitjançant Internet, ja que quan no es tingui accés a la xarxa es podrà seguir utilitzant l'aplicació sense problemes. Per altra banda, una partitura amb un nombre elevat de pàgines té un pes considerable i carregar-la a la RAM suposaria ocupar massa espai d'aquesta que, com s'ha dit, és escassa.

De cada arxiu es crearà una entrada a la base de dades de la memòria interna gestionada per CoreData[6]. En aquesta entrada s'hi desaran per una banda el **nom** del fitxer (que en el cas que estigui repetit s'afegirà un -copy), el nombre total de **pàgines** que té i el **path** on es troba (ubicació dins la memòria interna). Per altra banda es desaran un seguit d'informacions relacionades amb l'**estat** de producció d'aquella peça, el nombre **màxim** i **mínim** de *take* enregistrat i també la **pàgina** i ***take*** actual (per tal de poder obrir el document per on l'usuari l'havia deixat l'última vegada que hi havia treballat). Aquesta informació desada primerament a la memòria interna del dispositiu, pot ser utilitzada a l'aplicació en diverses ocasions i per tant, es passarà a la RAM en diferents moments. Els noms dels fitxers, per exemple, s'utilitzaran per mostrar-los a la secció *home* en forma de taula, on l'usuari podrà seleccionar-ne un per accedir-hi. Per altra banda, el nombre màxim o mínim de *take* s'utilitzarà a la secció "REC"per saber quan una presa és nova o bé ja hi ha anotacions fetes. Totes aquestes informacions derivades pròpiament de la producció, s'han anat afegint a la base de dades quan s'ha necessitat i és molt probable que a mesura que avança el projecte se n'afegeixin més o es

[6] `https://developer.apple.com/reference/coredata`

redissenyin les actuals.

Mentre es visualitza l'arxiu per pantalla, per poder accedir ràpidament a les pàgines anteriors o posteriors més properes i no haver d'esperar que s'acabi de carregar la imatge, es carregaran algunes pàgines del PDF a la RAM creant un petit buffer. D'aquesta gestió del buffer i tot el relacionat amb la visualització del PDF s'encarrega la llibreria que s'ha instal·lat abans de començar a programar i per tant, a priori no ha de ser una preocupació.

4.2.2 Classes pròpies

Un cop l'usuari hagi obert una partitura i vulgui començar a anotar-hi coses s'hauran d'anar desant totes aquelles anotacions que vagi fent. Aquestes anotacions tindran diversos paràmetres que serviran posteriorment per a treballar amb les dades. Per a determinar els paràmetres necessaris que caldrà desar s'ha mirat sempre el global del projecte i quines són les dades que es podrien necessitar per a fer cadascuna de les funcions que, un cop acabada, es voldrà que faci l'aplicació. Tot i així, igual que passa amb la base de dades dels arxius, s'hi van afegint dades a mesura que el projecte ho necessita.

Per una banda s'haurà de poder determinar quin tipus de **nota** és i en quina presa (***take***) ha estat anotada. Dintre dels tipus de nota s'establiran en un primer moment 8 diferents on 0 és una anotació de canvi de sistema i les anotacions de l'1 al 5 seran de menys a més puntuació musical. Aquest paràmetre també es farà servir per determinar anotacions d'inici o final de presa (98 i 99). Per altra banda, caldrà saber en quina posició de la partitura ha estat anotada i per tant es necessitarà la **pàgina** i la posició exacta de la ***x*** i la ***y***.

Coneixent aquests paràmetres es pot determinar on va situada cada anotació i quin tipus de símbol l'acompanya. Per a pintar-ho en pantalla però, falta l'element bàsic, el **botó**. El botó és un tipus de classe interna de Swift (UIButton[7]) que té moltes funcions incorporades i seran molt útils quan per exemple es vulgui prémer aquella anotació, canviar-la de lloc, canviar-la de color, etc...

Finalment, per poder treballar més endavant amb aquestes dades, s'ha cregut necessari desar també el **sistema** en el que es troba (en relació a la pàgina actual) i l'**ID** d'aquella nota, el qual serà proporcionat per CoreData una vegada es desi l'entrada.

Totes aquestes dades es desaran tant a la memòria RAM com a la interna de la següent manera: Un cop l'usuari introdueixi una nova anotació, aquesta es desarà a un array (RAM) de la classe Notes (Figura 4.3) que a la vegada crearà una nova entrada a la base de dades. Al CoreData s'hi desaran les mateixes dades que s'han desat a la RAM excepte el botó, ja que s'haurà de crear només quan es vulgui veure per pantalla aquella anotació. Per altra banda, al CoreData s'hi especificarà l'**arxiu** on la nota en qüestió s'ha creat.

Quan es carrega un arxiu a l'aplicació, el primer que farà la funció ViewDidLoad[8] serà descarregar-se de la memòria interna (CoreData) a la RAM totes aquelles dades relacionades amb l'arxiu carregat. Com tota aquesta informació s'estarà desant a un array, la posició d'aquest array servirà per accedir a tota la informació sobre aquella nota de manera molt més ràpida tal com es detalla en el següent exemple:

Per poder canviar de lloc una nota s'ha de prémer el botó d'aquesta durant un segon. Internament, Swift té una funció[9] que havent-la implementat al botó a la seva declaració, s'activa quan detecta que s'ha premut el botó en qüestió durant un cert temps. Ara bé, la funció s'activarà sempre que qualsevol botó hagi estat premut durant el temps establert. Com s'ha explicat abans, totes les anotacions són botons i per tant, es necessitarà saber quin dels botons ha estat premut. Quan es crea un botó nou, a la seva declaració s'hi estableixen una sèrie de paràmetres bàsics com el color, la mida, la imatge,

[7] `https://developer.apple.com/reference/uikit/uibutton`
[8] `https://developer.apple.com/reference/uikit/uiviewcontroller/1621495-viewdidload`
[9] `https://developer.apple.com/reference/uikit/uilongpressgesturerecognizer`

la posició, etc... però també s'hi poden establir altres paràmetres de molta utilitat com el *tag*. Aquest tag és un número que serà accessible sempre que s'interaccioni amb el botó al qual ha estat assignat, en aquest cas, quan s'ha premut més d'un segon i s'activa la funció per canviar-lo de lloc. D'aquesta manera, si cada botó té un tag diferent, sabent el tag es podrà conèixer quin és el botó premut. La gràcia en aquest punt és que aquest tag s'ha fet coincidir amb el número de posició de l'array i per tant, cada vegada que es premi un botó es podrà accedir a tota la informació relacionada amb aquest.

Aquesta manera de relacionar els botons amb l'array (assignant el número de posició com a tag), té els seus handicaps. Quan es vol esborrar una anotació, l'opció lògica seria eliminar l'entrada de CoreData i també la de l'array (RAM) tal com s'ha fet al crear-la però, què passaria si l'anotació que s'està eliminant no és l'última de l'array? Totes les anotacions posteriors a l'eliminada passarien a tenir una posició menys a la que tenien inicialment i per tant, ja no correspondria al tag d'aquell botó. La solució trobada per resoldre aquest problema ha estat tenir una variable booleana[10] indicant si aquell botó és **visible** o no. Quan l'usuari elimina una anotació, internament només es suprimeix l'entrada de la CoreData però no la de la RAM, on simplement es canvia la variable *visible* a false. Després de suprimir es refrescarà la pantalla i com la nota eliminada portarà la variable *visible* en fals ja no es veurà. Quan es canvii de fitxer o es tanqui l'aplicació i s'hagi de tornar a crear l'array de la RAM, aquella anotació no apareixerà, ja que no es descarregarà de CoreData on sí que s'havia eliminat.

```
class Notes
{
    var pag = Int16()
    var sistema = Int16()
    var take = Int16()
    var nota = Int16()
    var x = Float()
    var y = Float()
    var id = String()
    var boto = UIButton()
    var visible = true
}
```

Figura 4.3: Declaració de la classe "Notes"

[10] https://developer.apple.com/reference/swift/bool

4.2.3 Paràmetres d'usuari

Per últim serà necessari desar tots aquells paràmetres assignables per l'usuari com poden ser el color, la mida de les anotacions o d'altres configuracions bàsiques que puguin sorgir. Per a desar aquests paràmetres no farà falta crear una base de dades de CoreData com s'ha fet anteriorment sinó que s'utilitzaran els anomenats UserDefaults[11] que, com el nom indica, desaran les configuracions per defecte de l'usuari a la memòria interna. Així doncs, si un usuari vol personalitzar l'aplicació de la manera que li sigui més còmode treballar, podrà fer-ho una vegada i la personalització quedarà desada fàcilment.

Com s'ha vist reflectit a l'enquesta, cada usuari té una manera de fer molt particular així que aquesta opció no és només una qüestió d'estètica com podria ser canviar el fons d'escriptori d'un ordinador, sinó que com més personalitzable sigui l'aplicació, a més persones podrà arribar i més èxit tindrà.

4.3 Procés d'implementació

Després d'haver organitzat tota la informació que es necessitarà mentre s'utilitzi l'aplicació, farà falta implementar totes les funcions. És a dir, passar totes les idees a codi i aconseguir que tot funcioni de la manera més ràpida i eficient possible.

En primer lloc es va implementar l'API de Dropbox per tal de poder obtenir les partitures de l'usuari. Per fer-ho es va haver de consultar la documentació que proporciona la mateixa empresa a la seva web per desenvolupadors.[12] Allà hi ha algun petit tutorial amb el qual seguint els passos es pot arribar a descarregar i/o pujar documents. Tot i així, la interfície gràfica per navegar per les carpetes, seleccionar un o més arxius, etc... s'ha hagut de programar tota de zero, ja que l'únic element que proporciona l'API és l'inici de sessió. Si es té l'aplicació de Dropbox instal·lada al mateix dispositiu, aquesta s'obre amb un missatge de confirmació per donar accés als teus arxius,

[11] `https://developer.apple.com/reference/foundation/userdefaults`
[12] `https://dropbox.github.io/SwiftyDropbox/api-docs/latest/`

per contra fa posar logIn i password a través del navegador.

Una vegada es va tenir accés als documents, es va haver de crear un petit gestor per navegar-hi (el que serà la secció *home*). Al prototip, aquest gestor ha estat una simple taula on es poden visualitzar tots els documents descarregats fins al moment, eliminar-los o bé clicar-hi per obrir-los.

Arribats a aquest punt l'aplicació era capaç d'entrar a Dropbox, descarregar-se un seguit de documents, tenir-los organitzats en una taula i mostrar-los per pantalla amb la llibreria prèviament instal·lada. És a dir, un visor PDF corrent. És a partir d'aquí que comença el desenvolupament propi de l'aplicació i al que cal parar-hi més atenció. El codi complert a dia 9 de maig de 2017 es pot trobar a la pàgina 96 de l'Annex. A continuació, es mostraran un seguit d'exemples d'algoritmes (un de cada secció principal) que s'han hagut de dissenyar per poder realitzar les funcions ideades:

4.3.1 Separació de sistemes:

Tal com s'ha explicat a la secció *Metodologia de programació* del capítol 3, abans de començar a prendre notes era molt important poder saber a quin sistema corresponen per tal de determinar si una nota va abans o després en el temps (musicalment parlant). Per poder-ho determinar s'hauran d'haver separat els sistemes a la preproducció, és allà on hi haurà un botó que clicant-lo posarà el document en *mode edició de sistemes*. Utilitzant la classe UITabGestureRecognizer[13] es poden establir funcions que es cridaran quan l'usuari toqui la pantalla el número de vegades que s'hagi establert a la declaració. En aquest cas, s'haurà de tocar la pantalla dues vegades per inserir una nova línia divisòria.

Quan l'iPad detecta que l'usuari ha tocat dues vegades la pantalla, després de comprovar que està en *mode edició de sistemes*, apareix una nova línia com aquesta[14] allà on s'ha tocat la pantalla. Un cop inserida es pot acabar d'ajustar la posició movent-la

[13] `https://developer.apple.com/reference/uikit/uitapgesturerecognizer`

[14] 

o bé eliminar-la deixant la pantalla premuda durant un segon.

Inserint aquestes línies, visualment ja es pot veure la separació dels sistemes però internament simplement és una qüestió estètica. Per poder realment separar la pantalla en tantes parts com sistemes tingui la pàgina, s'emmagatzemaran en dos arrays els límits de cada zona (un array pels mínims i l'altre pels màxims). El primer pas és diferenciar quan s'està separant per primer cop la pantalla (a l'introduir la primera línia divisòria) o bé si ja hi ha alguna separació. Si és la primera vegada, es crearan dos nous índexs als arrays, un amb el mínim a 0 i el màxim a la y del punt on es vulgui dividir, mentre que el segon índex tindrà com a mínim la y del punt i com a màxim un número més gran que el nombre de píxels que té la pantalla (9999 per exemple). D'aquesta manera ja es tindran separats el primer i el segon sistema. Quan s'introdueix una nova separació, tant el màxim de l'índex anterior com el mínim de l'índex nou passen a ser el punt y tocat i el màxim del punt nou passa a ser 9999.

Al moment d'introduir una anotació nova s'utilitzaran aquests marges per saber dins de quin sistema es troba. Fent servir la variable *range* del tipus Range[15] es podrà determinar el marge al qual pertany el punt y de la nova nota. A la Figura 4.4 s'hi pot veure l'algoritme que, simplement sabent l'índex de l'array on s'ha trobat que hi encaixa el marge, pot saber quin és el sistema de l'anotació (sumant-ne un perquè el primer sigui 1 i no 0).

```
func quinSistema(_ punt:Float) ->Int16
{
    var sistema = 0
    for n in 0 ..< sMax.count
    {
        let range:Range = Float(sMin[n])..<Float(sMax[n])
        let iHaveIt = range.contains(Float(punt))
        if (iHaveIt == true)
        {
            sistema = n+1
        }
    }
    return Int16(sistema)
}
```

Figura 4.4: Funció per determinar el sistema d'una anotació

[15] https://developer.apple.com/reference/swift/range

Ara bé, tal com està dissenyat l'algoritme, si la segona divisió es fes per sobre la primera, si s'elimina una divisió intermèdia o bé es canvia de lloc alguna altra, el *range* podria tenir un mínim més gran que el màxim i, al provar de fer la comparació, l'aplicació es penjaria. Per salvar aquest problema, cada vegada que es crea, s'elimina o es mou un nou sistema, es refan tots els marges reordenant les y de totes les separacions per tal que els sistemes quedin ordenats igual que les y emmagatzemades.

Sabent sobre quin sistema s'està tocant la pantalla ara ja es poden començar a introduir anotacions:

4.3.2 Entrada d'anotacions:

Totes les anotacions que es vagin prenent aniran relacionades amb un número de presa tal com es fa amb el protocol d'enregistrament original. Per tant s'ha hagut de dissenyar un marcador on diu la presa actual i es pot avançar o retrocedir d'una en una. Al disseny original aquest marcador sempre començava a la presa número 1 però, després d'analitzar els resultats de l'enquesta, aquest número haurà de poder-se introduir manualment deixant premut el marcador durant més d'un segon.

En començar una nova presa la primera anotació que s'haurà de crear serà el punt d'inici. Com sempre serà la primera cosa que s'anotarà, per tal de no perdre temps, el primer clic que es faci després de canviar de presa determinarà el punt d'inici, sense haver de prémer cap botó ni res. Un cop marcat l'inici es podrà començar a prendre notes pròpiament de l'enregistrament. La manera que s'ha cregut més ràpida per introduir noves notes és simplement tocant la pantalla un nombre determinat de vegades. En el prototip per exemple, si es toca una vegada, apareix el número de presa ratllat, al tocar-la dues vegades, apareix amb un +/-, tres vegades correspondria a un *ok*, quatre que la presa és bona i cinc que la presa és extraordinàriament bona. Una de les millores que caldria fer és poder diferenciar quan s'està tocant la pantalla amb un o dos dits, d'aquesta manera es podria arribar a tenir 10 símbols predeterminats diferents, cinc quan es toca amb un dit i cinc amb dos. Ara per ara no hi ha l'opció de canviar de símbols predeterminats, però en un futur també s'haurien de poder introduir manualment en algun lloc de la configuració.

Quan l'aplicació detecta que s'ha tocat la pantalla una vegada, activa la funció corresponent que, després de comprovar que la secció actual és "REC", inserirà una nova nota al lloc on s'ha tocat i desarà tota la informació com s'ha comentat a la secció anterior. Per fer aquesta anotació es cridarà la funció *drawNote* on es crearà el botó amb tots els atributs necessaris (el tag que s'ha parlat abans, el color, el símbol corresponent, la classe UILongPressGestureRecognizer, etc...). Per poder mostrar el número de presa a cada anotació, el *label* d'aquest botó serà la variable *currentTake*, és a dir, la mateixa que mostra el número de presa al marcador superior. La funció cridada quan l'aplicació detecta que s'ha tocat la pantalla una o diverses vegades, és crida tan bon punt ho ha detectat, és a dir que per cridar la funció dels 5 tocs, abans s'hauran cridat les altres quatre. Per evitar que no es creïn 5 anotacions superposades, les funcions de 2 a 5 tocs abans de crear una nova nota eliminen l'anterior.

Un cop s'ha creat la presa nova, es podrà canviar per una altra mitjançant uns botons que s'activaran a la barra superior i que permetran canviar el tipus d'anotació (dins les predeterminades), moure-la de lloc o esborrar-la tal com es pot veure a la Figura 4.5. Si es vol canviar, moure o eliminar una anotació que no és l'última, simplement s'haurà de deixar premuda durant un segon i s'activaran les mateixes icones però en un altre color.

Després d'haver pres les notes corresponents, s'haurà de marcar el punt on hi ha el final de presa. Tot i haver-ho redissenyat després de l'enquesta i no ser un paràmetre obligatori, sí que s'emetrà un avís si s'intenta canviar de presa sense haver-lo marcat. Més que res, és molt importat conèixer aquest punt per després poder crear els algoritmes

Figura 4.5: Barra superior a la secció REC

de la secció "POST". Si el final està marcat o bé es passa per alt l'avís, s'haurà de prémer el botó +, el número del marcador canviarà i es repetirà tot el procés anterior.

Per facilitar la lectura de les anotacions i els canvis/moviments que es vulguin fer, només estaran actius els botons de les anotacions de la presa actual, d'aquesta manera les preses antigues es veuran amb menys opacitat i no seran accessibles al tocar-les, simplement seran mostrades per pantalla.

4.3.3 Preses a un punt

Una de les funcions més senzilles, però molt útil, i que abans no es podia fer amb paper, és poder saber quines preses hi ha en un punt concret de la partitura. La qüestió és trobar totes les preses que tinguin l'inici abans del punt tocat i que el final de presa sigui més tard. Així doncs, quan s'ha tocat la pantalla durant més d'un segon, s'activa una funció on després de desar el punt x tocat i mirar a quin sistema pertany l'y, hi ha un `for` que recorre totes les anotacions d'aquell document (Figura 4.6). Primer de tot es buscaran les anotacions que siguin un inici de presa i d'aquestes es desaran totes les que aquest inici sigui abans del punt tocat.

```
for n in 0..<docNotes.count
{
    if (docNotes[n].nota==98) //Busco les anotacions d'inici de presa
    {

        if (docNotes[n].sistema < sistemaPunt || (docNotes[n].sistema == sistemaPunt && docNotes[n].x < Float(punt.x)))
        {
           possiblesTakes.append(docNotes[n].take) //Em quedo les preses que comencen abans que el punt tocat
        }

    }
    for i in 0..<possiblesTakes.count
    {
        //print("Numero de i: " + String(possiblesTakes.count))
        if (docNotes[n].nota==99 && docNotes[n].take==possiblesTakes[i]) //Busco el final de presa dels takes que m'he quedat
        {
            if (docNotes[n].sistema > sistemaPunt || (docNotes[n].sistema == sistemaPunt && docNotes[n].x > Float(punt.x)))
            {
                print(docNotes[n].take) //Em quedo amb els que acaben més tard que el punt
            }
            else
            {
                possiblesTakes.remove(at: i) // I elimino els que acaben abans
            }
        }
    }

    }
```

Figura 4.6: Loop per trobar les preses a un punt de la partitura

Seguidament i encara dins del primer loop, se'n fa un altre tantes vegades com preses s'hagin desat anteriorment per mirar si aquella anotació té un final i, en cas de ser-hi, estigui més tard que el punt tocat. Si és així, aquella presa es mantindrà dins l'array però, si el final és abans del punt en qüestió, aquella presa s'eliminarà. D'aquesta manera, quan hagi acabat el loop principal, a l'array només hi quedaran les preses que comencin abans i acabin més tard que el punt.

4.4 Prototipatge

Per poder anar implementant totes les funcions i veure la funcionalitat de l'aplicació es necessita un mínim disseny gràfic que sigui molt flexible a l'hora de poder afegir o treure elements fàcilment. És per això que s'ha hagut de crear un prototip basat en el mockUp[16] però accessible des de l'Xcode per poder-lo fer interactuar amb el codi.

4.4.1 StoryBoard:

El primer pas ha estat crear l'StoryBoard de l'aplicació, és a dir, dibuixar totes les possibles pantalles que podrà haver-hi (*home*, PRE, REC, POST, etc...) i enllaçar-les entre elles mitjançant botons. Des de l'eina gràfica d'Xcode ha estat bastant senzill fer-ho, ja que simplement s'han de crear unes pantalles anomenades ViewController[17] i arrossegar-hi els elements necessaris. Després, per fer els enllaços, es prem la tecla *Ctrl* i s'arrossega una línia des d'un botó fins a la pantalla que es vol que mostri al ser clicat.

Com es pot veure a la Figura 4.7 cadascuna de les pantalles té diversos *fils* que entren i surten. Això és degut al fet que des de cada pantalla es pot accedir a qualsevol de les altres tres i per tant, es necessitarà un fil per a cada enllaç. S'ha anat comentant al llarg del treball que l'aplicació consta de 4 pantalles principals tot i així, no ha de sorprendre que a l'strroryBoard s'hi vegin unes quantes més. Les que tenen una gran part de color blau clar són les seccions principals (PRE, REC i POST) i aquesta part blava que

[16]Veure secció 3.1 Esquema i MockUp.

[17]`https://developer.apple.com/reference/uikit/uiviewcontroller`

es veu és precisament la partitura. Les dues pantalles grises que es veuen al centre de la captura són el que s'ha anomenat *visor*, ja que és el ViewController on es veurà el PDF amb la llibreria corresponent. Per no haver de crear un *visor* a cada pantalla, només s'ha creat aquest que està inserit a la zona blava de les pantalles principals. D'aquesta manera, les diferents seccions només controlen la barra lateral/superior i les funcions pròpies d'aquella secció, mentre que les funcions comunes s'han de programar una sola vegada al *visor*.

A la part de sota la captura es pot veure la secció *Home* on la part grisa és la taula que s'omplirà amb els arxius que tingui l'usuari. Per altra banda, es pot veure el popUp de Dropbox que apareixerà quan es toqui el botó corresponent. Quan s'instal·lin més APIs de diferents serveis hi haurà una pantalla com aquesta per a cadascun d'ells.

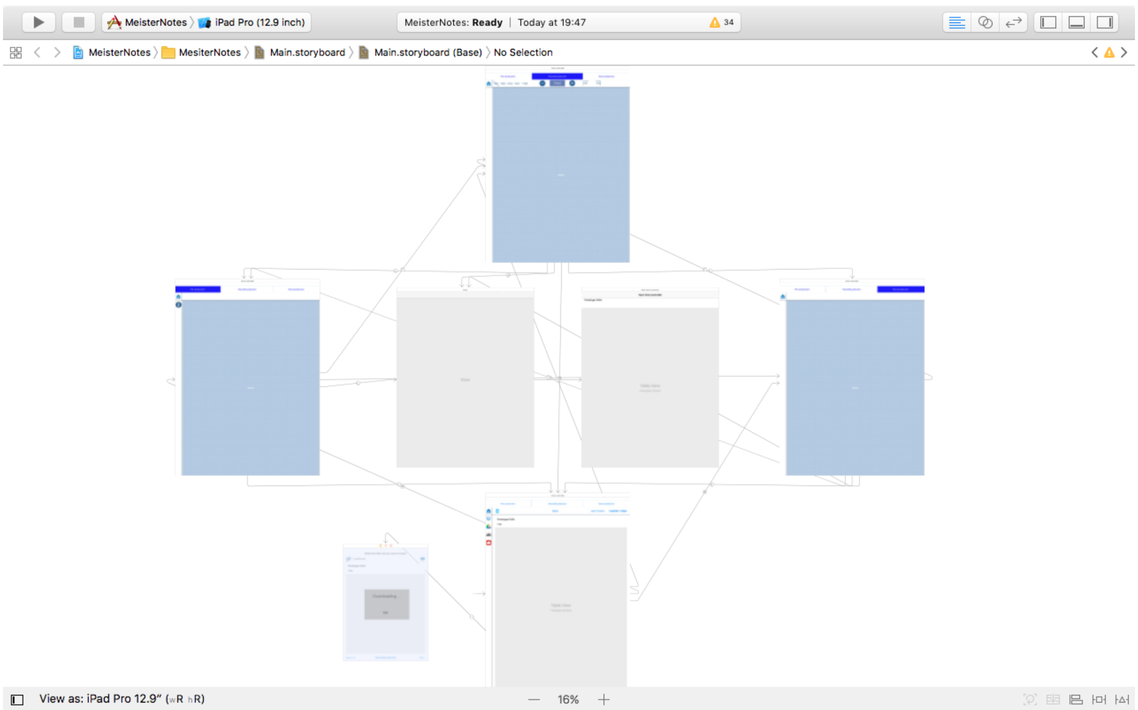


Figura 4.7: StoryBoard de l'aplicació

4.4.2 Disseny del prototip:

A mida que l'StoryBoard ho anava necessitant s'ha anat creant un petit disseny gràfic que permetés fer-se una idea de com serà el producte final. Bona part d'aquest disseny s'ha fet de la manera més senzilla possible, ja que en qualsevol moment del procés d'implantació es pot necessitar algun canvi i no val la pena perdre molt de temps fent un gran disseny si després no serveix per a la funció que ha de servir. Per tant, tot el que són les barres superior i laterals, així com també els botons de canvi de secció s'han fet directament des de l'Xcode posant uns *label* amb el fons de color blanc i sense cap text (el resultat acaba sent un rectangle del color del fons). També s'han utilitzat els mateixos botons de canvi de secció que, pintant de blau el que està actiu i de blanc els altres dos permet donar una mica de dinamisme al disseny d'una manera senzilla.

El que sí que s'ha hagut de dissenyar expressament han estat les icones pròpies, és a dir, la barra divisòria de sistemes, el botó per activar la funció de separació de sistemes, els botons de + i - per canviar de *take*, les icones de les anotacions, etc... Aquestes icones s'han creat amb el software especialitzant en disseny d'aplicacions anomenat Sketch.[18] Per altra banda, les icones dels serveis d'emmagatzematge s'han descarregat d'Internet de manera provisional per a poder fer el prototip.

A la pàgina 137 de l'Annex, s'hi poden trobar captures de pantalla del prototip. És interessant també, poder veure l'evolució del mockUp inicial al disseny del prototip. Fent la comparació, es pot veure que el disseny principal s'ha mantingut bastant, però hi ha hagut petites modificacions degudes a l'evolució del projecte.

Com ja s'ha comentat anteriorment, però és important remarcar-ho, en tot moment s'ha intentat perdre el mínim temps possible per a fer el disseny gràfic del prototip. És un disseny que va evolucionant i l'important no és l'estètica sinó la funcionalitat i capacitat d'adaptació. Un cop el prototip ja estigui tancat, testat i amb poc marge de variació, aleshores sí que s'encarregarà un disseny nou a un professional.

[18] https://www.sketchapp.com/

4.5 Avaluació

Aquesta part és molt important per saber si tot allò que s'ha fet funciona tal com s'ha previst que ha de funcionar. És a dir, que algú que veu per primera vegada l'aplicació acabada és capaç d'aprendre'n el funcionament i utilitzar-la amb fluïdesa. Per fer l'avaluació del prototip se seguiran els passos que proposa Neo Monefa a la seva guia de desenvolupament per a principiants (Monefa (2015)).

Per realitzar aquestes proves es necessitarà la col·laboració de gent aliena al projecte però relacionada amb el sector a qui se li deixarà provar i explorar l'aplicació durant una estona. Després, se'ls hi demanarà que indiquin quins són els punts forts i els punts febles que han trobat i es discutiran els temes que sorgeixin entre els participants. Si costa treure temes de discussió, es poden dur preparades algunes preguntes per intentar desencallar la situació i treure el màxim d'informació sobre la prova. Per exemple es pot preguntar si s'han sentit còmodes utilitzant l'aplicació, si s'han complert les expectatives que s'esperaven sobre una app com la que s'ha provat o per contra si s'ha trobat a faltar alguna cosa, la usabilitat de la interfície gràfica (en aquest cas el prototip encara), etc...

Monefa també diu que és important que la postura del desenvolupador sigui el màxim de neutre possible per deixar expressar totes les opinions possibles. En aquest cas, per exemple, és molt probable que qui provi l'aplicació trobi a faltar algunes funcions que ja estan incloses al disseny però no s'han implementat al prototip. La primera reacció podria ser explicar-ho i dir que ja s'implementarà però, d'aquesta manera, es pot arribar a cohibir als participants de la prova fent que deixin de dir alguna cosa pensant que ja està contemplada. Per tant, si es manté una posició neutra, també es podrà veure quines de les funcions contemplades en el disseny es troben més a faltar i quines altres no corren tanta pressa.

CAPÍTOL 5

Passos a seguir després del prototip

5.1 Disseny gràfic

Un cop l'aplicació estigui pràcticament acabada i sigui difícil que hi hagi grans canvis que puguin alterar-ne el disseny, serà el moment de fer el disseny gràfic definitiu. És a dir, aquell disseny amb el qual es presentarà públicament i es posarà a la venda l'aplicació. Arribats a aquest punt s'ha de prendre una decisió sobre si seguir desenvolupant l'aplicació individualment i fer-ne també el disseny, o bé encarregar-lo a un tercer. Cadascuna de les opcions té els seus avantatges i inconvenients:

5.1.1 Disseny propi:

Segurament semblarà l'opció més econòmica i la que més ràpid s'adaptarà a l'aplicació, però no només per això ha de ser la millor opció. Per fer un bon disseny gràfic no hi ha prou amb tenir un mínim de coneixement de les eines, sinó que s'han de conèixer el suficient per poder fer tot allò que es vol fer. A banda, també es necessita tenir un mínim talent artístic per poder fer un disseny atractiu, que entri per la vista als usuaris, però que a la vegada sigui suficientment funcional. En aquest cas hi hauria dues opcions:

- **Dissenyar-lo de zero:** Dissenyar-lo des de zero voldrà dir tenir una idea al cap molt clara de com es vol que sigui aquest disseny i, després d'estar veient durant mesos el disseny del prototip, el més probable és que acabi quedant un prototip millorat, que no ha de ser dolent per força, però segurament no serà igual de "vistós"que un disseny nou.

  En aquest cas es pot utilitzar un software especialitzat o bé seguir utilitzant l'eina gràfica d'Xcode que, a banda de les virtuts que ja s'han comentat, també conté moltes icones de la UI pròpia d'Apple. Si s'utilitzen bé es podria arribar a crear una aplicació d'estètica semblant a les aplicacions natives que porta iOS.

- **Utilitzar plantilles:** Seguint en la línia de crear una aplicació mitjançant icones predissenyades, una altra opció seria usar plantilles o kits de tercers. Aquesta seria una opció intermèdia entre un disseny propi i el d'un tercer.

  Hi ha empreses que proveeixen kits per poder dissenyar apps sense haver de començar de zero i amb un atractiu visual important. En aquest cas però, el problema serà adaptar aquestes plantilles a l'app que s'està desenvolupant. A tall d'exemple, fent una cerca ràpida a google, s'han trobat kits com el Tethr[1], SketchAppSources[2] o Angle[3]. Si es troba una plantilla que té una similitud a la idea original de l'aplicació, aquesta seria una opció econòmica i relativament fàcil d'adaptar, ara bé, si en cas contrari no es troba cap plantilla que s'assembli, es pot perdre molt de temps intentant adaptar-ne una altra i el resultat pot no ser massa bo.

En definitiva, si es té molt clar el disseny a fer i certa gràcia artísticament parlant, dissenyar l'aplicació un mateix és una bona opció, però si no és el cas, intentar-ho pot fer perdre molt de temps i el resultat difícilment serà tan bo com si es contracta un professional. Si es tria aquesta opció per qüestions econòmiques, val la pena pensar quin

[1] `https://www.invisionapp.com/tethr`
[2] `https://www.sketchappsources.com/category/ui.html`
[3] `https://designcode.io/angle/`

preu tenen les hores que s'hi estan invertint personalment i si, després de contar-les, segueix sent més econòmic que contractar un tercer.

5.1.2 Disseny d'un tercer:

Tot i ser l'opció més cara, econòmicament parlant, sens dubte serà la que més bon resultat final pot arribar donar i segurament menys temps farà perdre, sempre que s'esculli el dissenyador adequat. De dissenyadors hi ha molts i triar-lo no és una qüestió fàcil. Primer de tot s'ha de decidir si es contractarà via online o bé presencialment. En tots dos casos s'haurà de triar un havent-ne vist unes quantes mostres de feines fetes anteriorment i en aquestes, poder-hi visualitzar un possible disseny de l'aplicació pròpia.

- **Contractació online:** Aquesta opció té molts avantatges, el més important segurament és que hi ha moltíssima més oferta i per tant, serà més fàcil trobar un dissenyador que s'adapti a les necessitats de l'app i la butxaca. Per una banda existeixen webs on es poden trobar ofertes de dissenyadors d'arreu del món com Toptal[4], Upwork[5] o Fiverr.[6] En aquestes webs (sobretot a l'última esmentada) hi ha ofertes per tots els gustos, colors i preus, per aquest motiu és important triar bé l'adequada i mirar què entra en el preu que ofereixen, ja que moltes vegades només hi entra una sola revisió de la feina, és a dir, que si pel que sigui la primera feina que es rep no acaba de ser l'esperada, només es té una opció a rectificació i si tot i així continuen havent-hi aspectes a millorar, s'haurà de pagar un suplement. Per tant, val més buscar una oferta que encara que d'entrada sigui més cara, tingui més opcions a revisió. Per altra banda, es poden buscar dissenyadors gràfics concrets fent una simple recerca o mitjançant recomanacions d'altra gent.

[4]https://www.toptal.com
[5]https://www.upwork.com/
[6]https://www.fiverr.com/

- **Contractació presencial:** Si s'opta per contractar algun dissenyador presencialment, a banda de fomentar els negocis de proximitat, en general es tindrà una comunicació més propera i serà més fàcil explicar les idees, funcionament de l'app, etc... És més, molt probablement, un cop el dissenyador tingui els primers esborranys, es podrà anar al seu estudi i entre els dos acabar de fer els retocs pertinents.

5.1.3 Formalització

Si finalment s'acaba optant per l'opció d'encarregar el disseny gràfic a un tercer, tant si és via online com si es fa presencialment s'hauran de preparar un seguit de documents per tal que el dissenyador pugui fer el disseny el més acurat possible i més proper a l'aplicació. Cada dissenyador pot demanar més o menys coses, però és important que se li passi el màxim d'informació possible per poder obtenir bons resultats. Tot i així, hi ha un seguit d'aspectes que la majoria de dissenyadors és molt probable que demanin:

- **MockUp de l'aplicació (o prototip si es té):** Es necessita per poder treballar sobre un disseny inicial propi del desenvolupador. En el cas que s'està tractant en aquest treball, se li diria que mentre els elements que es mostren al prototip/mockUp també estiguin visibles al disseny final, és lliure de canviar algunes coses de lloc sempre i quan sigui per millorar-ne l'accessibilitat. És a dir, si en comptes d'una barra lateral vol posar una barra inferior o unes bombolles flotant, no hi hauria problema.

- **Mostra del funcionament:** En el cas que es tingui el prototip funcionant, com serà el cas, s'enviarà també una mostra de l'aplicació en funcionament. D'aquesta manera el dissenyador pot veure quines són i com són les funcions que fa i pot tenir alguna idea per millorar la comunicació usuari-app.

- **Explicació del funcionament:** Es tingui o no la mostra també demanaran que s'envii l'explicació per escrit del funcionament de l'aplicació, la finalitat que té cadascuna de les pantalles i el contingut que s'hi veurà (en aquest cas: partitures).

- **Descripció de l'empresa i el target de clients:** Per tal d'adaptar el disseny als estàndards del sector i no fer una aplicació massa cridanera per a un públic més aviat seriós o a l'invers.

- **Inspiració i/o competència:** Havent fet l'estudi de mercat segur que s'ha trobat alguna aplicació amb un disseny que podria inspirar la pròpia. En aquest cas no es té competència directa pel que fa a la funcionalitat de l'aplicació, però sí que servirien com a inspiració altres aplicacions per llegir partitures.

- **Altres qüestions que puguin ajudar:** Molts dissenyadors també demanen la icona de l'aplicació (si es té), preferències de colors, si agrada més un disseny minimalista, més carregat, o qualsevol altra informació que pugui ser rellevant per a fer un disseny més a mida.

Havent enviat totes aquestes informacions al dissenyador, un cop estigui el disseny donat per bo, es rebrà en el format acordat prèviament. Majoritàriament el dissenyador enviarà el material en arxius d'Adobe Photoshop[7] o bé d'Sketch[8]. Tots dos softwares permeten treballar per capes, fet que permetrà tenir tots els elements (botons, barres, etc...) per separat i poder implementar-los al codi amb relativa facilitat.

5.1.4 Implementació

Rebuts els documents per part del dissenyador, caldrà traslladar el nou disseny gràfic a Xcode per tal que interactuïn amb el codi. Per una banda hi ha els elements que es podrien anomenar *comuns* a totes les apps com són els botons de text, els textos, etc... Aquests elements s'hauran de traslladar del disseny a l'app mitjançant l'eina gràfica d'Xcode de la que s'ha parlat anteriorment. Simplement es crearan aquests elements i s'ajustaran les mides, fonts o colors que s'han rebut del dissenyador. A tota aquesta

[7] `http://www.adobe.com/es/products/photoshop`
[8] `https://www.sketchapp.com/`

informació s'hi podrà accedir amb l'Sketch mateix o amb algun plugIn que faciliti la feina com Zeplin.[9] Hi haurà casos en què l'eina gràfica d'Xcode es quedarà curta i no deixarà personalitzar prou algun element. Quan passi això s'haurà de crear una classe on es sobreescriuran algunes funcions internes (mitjançant la comanda `override func`, mai canviant el codi original que no és accessible). Això passarà per exemple per poder tenir una entrada de text amb el requadre a la vista, o per canviar el color d'algun element predefinit.

Per altra banda hi haurà elements del disseny que no serà possible crear-los des de l'eina gràfica com, per exemple, les imatges de fons, les icones d'alguns botons o altres imatges que puguin aparèixer. El procediment més habitual és exportar les diferents capes d'imatges rebudes en els fitxers d'Sketch i importar-les a Xcode. Si es fa d'aquesta manera, s'ha de tenir en compte que els diferents dispositius tenen diferents densitats de píxels a la pantalla i per tant, es necessitarà exportar cada imatge en tantes resolucions com a dispositius es vulgui arribar. A la figura 5.1 es pot veure els diferents dispositius i les resolucions que requereixen, en el cas dels iPad es necessitarà exportar en resolució original i en el doble de l'original. Aquesta exportació es pot fer directament des de l'sketch marcant la casella amb les diverses resolucions que es vol exportar. Si s'exporta així, automàticament es desaran amb els noms de fitxer *nomDelFitxer.png* per l'original i *nomDelFitxer@2x.png* per la versió el doble de gran. D'aquesta manera, a l'hora d'importar-ho a l'Xcode es podran importar totes les mides a la vegada, Xcode les organitzarà per nom (ometent l'@2x o @3x) i es podran implementar sense tenir en compte aquests canvis de mida. En el cas que l'aplicació tingui moltes imatges i que no es necessiti tenir alguna transparència, s'haurà d'intentar exportar sempre que sigui possible en format JPG, ja que al ser un format més simple que PNG, es carregarà més ràpid i l'app tindrà una millor fluïdesa.

Per no haver d'estar pendent de formats, resolucions, etc, hi ha l'opció de traslladar les imatges a codi, és a dir, escriure un codi que pinti els píxels dels colors que se li digui

[9] `https://zeplin.io/`

per tal de veure la imatge per pantalla. De fet, a molt baix nivell, encara que s'importi una imatge en PNG o JPG el dispositiu estarà fent això. Pot semblar una opció molt desbaratada i costosa però gràcies a certs plugIns no ho és gens. PaintCode[10] per exemple, permet exportar directament des d'Sketch a codi Swift. El fitxer generat contindrà una funció per cadascuna de les imatges que s'hagin traslladat i, al cridar-la, retornarà una UIImage[11] que es podrà fer servir de manera similar a la d'un fitxer d'imatge. La manera d'implementar-ho segurament és una mica més costosa que amb les imatges exportades, però pel que fa a fluïdesa és molt millor i tampoc s'ha d'estar pendent de resolucions estranyes.

En tots dos casos caldrà configurar els anomenats *constraints* per fer que el disseny funcioni amb les diferents mides de dispositiu. En el cas de l'iPad: l'iPad pro, l'iPad retina, l'iPad air i l'iPad mini. Els constraints consisteixen en un seguit de relacions entre objectes que permeten que el disseny correspongui quan l'aplicació s'utilitza amb un dispositiu o un altre. Les relacions poden ser entre un objecte i la pantalla completa (per exemple establint que hi hagi un element al centre de la pantalla, o que estigui sempre centrat verticalment o horitzontalment), poden ser entre un objecte i un altre objecte (per exemple dos botons que es vol que sempre mantinguin la mateixa distància lateral o vertical) o fins i tot un objecte respecte ell mateix (per exemple per mantenir

[10] `https://www.paintcodeapp.com/sketch`
[11] `https://developer.apple.com/reference/uikit/uiimage`

Device	Pixel Resolution	
iPad 1 & 2, iPad Mini 1, iPhone 3	1x	
iPad Air & iPad Retina & iPad Pro, iPad Mini 2 upwards, iPhone 4 upwards	2x	Retina
iPhone 6+, 6S+	3x	Retina HD

Figura 5.1: Font:`https://sitesforprofit.com/import-images-into-xcode-using-sketch`

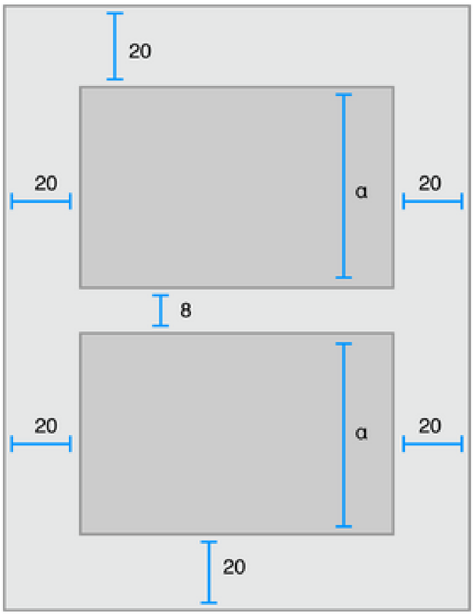


Figura 5.2: Font: https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/AutolayoutPG/

l'alçada o l'amplada encara que la mida de pantalla canvii).

Per acabar la implementació es poden posar animacions a les accions que fa l'aplicació, per exemple que es fongui al canvi de pantalla, que un popUp entri fent zoom, etc... Aquestes animacions es poden fer per codi (definint un estat inicial, un estat final i un temps de transició) o bé amb alguna llibreria amb efectes predissenyats com Canvas[12].

5.1.5 Disseny de la icona

El disseny de la icona es pot encarregar o fer un mateix en qualsevol moment del desenvolupament, ja que no afecta en absolut al funcionament de l'aplicació. En aquest cas, s'ha encarregat per poder-lo tenir a temps a la presentació del prototip. És important tenir un disseny de la icona que concordi amb la finalitat de l'aplicació, ja que a les botigues d'aplicacions serà el que la gent veurà abans de comprar-la. Per tant, en aquest cas necessita tenir un tret identificatiu com podria ser algun símbol del protocol d'enregistrament o bé de l'enregistrament de música clàssica en si.

[12]http://canvaspod.io/

Per dissenyar la icona de l'aplicació s'ha de tenir en compte tot el que s'ha parlat anteriorment en relació a la recerca de dissenyador. A més, quan s'encarregui aquest disseny també es necessitarà enviar informació semblant a la que s'ha enviat per a fer el disseny de la interfície gràfica, si més no, tot el relacionat amb el target a qui va dirigida l'app, la funcionalitat que té, preferències de colors i estil, etc...

5.2 Sortida a l'AppStore

Una vegada donada per bona l'aplicació, o si més no, la que serà la primera versió pública amb la que es donarà a conèixer, caldrà preparar-la per publicar-la a l'AppStore, la botiga d'aplicacions d'Apple. Així com Android permet pujar tota mena d'aplicacions sense haver de passar cap revisió, Apple és més estricte i exigeix complir un seguit de requisits abans de publicar qualsevol aplicació. És per aquest motiu que la majoria d'aplicacions tenen un aspecte similar. Molts d'aquests requisits són de caire estètic i estan explicats en una guia que proporciona Apple mateix[13] i si el dissenyador els ha seguit (que és el més normal) només s'hauran de tenir en compte la resta.

El primer requisit és haver realitzat proves suficients de l'aplicació mitjançant una eina a la que es té accés sent desenvolupador, i amb la qual es pot passar l'aplicació a 3rs sense que estigui encara publicada. Seguidament es necessitarà tenir un compte de desenvolupador d'Apple i haver pagat la subscripció anual a l'iOS Developer Program[14] (uns 100€). Un cop dins s'hauran de crear diversos certificats tant com a desenvolupador com per la pròpia app (IDs que relacionin l'aplicació amb el seu desenvolupador, drets de copyright, etc...). Després caldrà revisar la guia estètica que s'acaba de comentar i preparar el projecte d'Xcode, pujar-la a l'iTunes a l'espera que s'aprobi i finalment publicar-la a l'appStore.

[13] `https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/`

[14] `https://developer.apple.com/programs/`

No s'ha volgut profunditzar molt en aquest aspecte ja que apple ofereix una guia[15] on està tot explicat en detall i d'una manera molt entenedora. És més, qualsevol modificació dels passos a seguir, estarà sempre actualitzada a l'enllaç que hi ha al peu de pàgina.

5.2.1 Gratuïtat o preu de l'aplicació

Encara abans de llançar l'aplicació al mercat s'haurà de decidir si aquesta serà gratuïta o bé quin serà el seu preu. Tractant-se d'una aplicació amb la finalitat d'ajudar al productor musical i que, en definitiva, pot arribar a incrementar la productivitat i guanys d'aquest, es creu convenient fixar un preu a l'aplicació prou atractiu però tampoc massa baix. S'ha de tenir en compte també, que d'aquest preu, Apple se'n quedarà un 30%.

Tot i així no es descarta fer una versió gratuïta de l'aplicació amb algunes funcionalitats retallades. Aquesta versió podria servir per una banda per que la gent pugui provar-la abans de comprar la bona i així intentar reclutar a més públic. Per altra banda però, també pot servir perquè en conservatoris o universitats on es faci alguna assignatura semblant a la de *Tècniques d'enregistrament i postproducció I*, els alumnes puguin descarregar-se-la i testejar-la a classe sense haver de pagar. A les aplicacions gratuïtes se'ls hi pot introduir publicitat i rebre una remuneració a canvi, d'aquesta manera, es poden aconseguir uns ingressos extres.

5.2.2 Manteniment

Tal com s'explicava a la introducció d'aquest treball i s'il·lustrava la Figura 1.1 de la pàgina 7, el desenvolupament d'un software és una feina circular que si es vol estar actualitzat mai s'acaba. Per tant l'aplicació necessitarà un cert manteniment; per una banda per mantenir-se al dia de les diverses actualitzacions de sistema operatiu que aniran sortint i, per altra banda per anar afegint o millorant funcions de la pròpia aplicació.

[15] `https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppDistributionGuide`

Per exemple, una de les primeres coses que caldrà fer serà *netejar* el codi per incrementar-ne la fluïdesa seguint els passos del llibre *Clean Code* (Martin, 2009). És a dir, revisar el codi i intentar optimitzar les funcions de tal manera que hagin de fer menys operacions per acabar arribant al mateix resultat. També caldrà retocar qüestions que els primers usuaris s'hagin pogut anar trobant les primeres setmanes d'us.

CAPÍTOL 6

CONCLUSIONS

Després de tants mesos de feina costa escriure unes conclusions que resumeixin tota la feina feta i els obstacles que s'han hagut d'anar superant. Tot i així, la lliçó amb la qual es podria resumir bona part d'aquest procés és que tot i els obstacles que es puguin trobar, sempre s'ha de mirar endavant i intentar fer el que es vol fer de la manera que calgui. Si hi ha un aspecte que no deixa seguir pel camí que s'estava seguint, potser hi haurà un altre camí per on es podrà avançar i, més tard amb el cap fred, l'aspecte que semblava no tenir solució es podrà solucionar amb facilitat.

Poden semblar unes paraules molt vagues i generalistes però en el món de la programació, si no es tenen en compte, es pot acabar molt malament amb frustració rere frustració, ja que, aquest aspecte tan complicat potser és simplement una lletra equivocada o un canvi de variable on no tocava que fa fallar tota l'aplicació.

Un clar exemple ha estat el fet d'haver d'implementar una llibreria per llegir arxius PDF. Semblava la qüestió menys important de l'aplicació, ja que no era cap element diferencial, però alhora era imprescindible per poder desenvolupar tota la resta de funcions. Es van provar moltes llibreries i va costar molt trobar l'adequada. Tot i així, reprenent el que s'estava dient, hi va haver un punt on es va decidir seguir amb el projecte encara que la llibreria no funcionés del tot bé. Es tenia una llibreria on no es podia passar de pàgina d'una en una ni tampoc accedir a la funció que feia canviar de pàgina. Malgrat aquest inconvenient es van implementar tots els algoritmes de separació de sistemes, entrada d'anotacions, detecció de les preses en un punt, etc... treballant amb una sola pàgina. Això sí, sempre preparant el codi per quan es trobés una nova llibreria poder fer el canvi ràpidament.

Durant el desenvolupament de l'aplicació s'ha pogut veure la importància que té esquematitzar les coses des d'un primer moment. En el moment de fer el disseny funcional el primer pas va ser esquematitzar les idees inicials en un prezi i des de llavors, passant per un mockUp, el disseny de l'Storyboard i fins i tot a la implementació, s'han anat fent petits esquemes per tal de veure més clar què és el que s'estava demanant que l'aplicació fes. La idea pot estar molt clara al cap però el simple fet d'escriure-la en un paper i haver de forçar-se a decidir quin pas va abans i quin després, ha ajudat molt. De fet, s'ha omplert gairebé tota una llibreta amb dibuixos, esquemes, passos a seguir d'un algoritme, etc...

A l'inici d'escriptura de la memòria s'ha elaborat una enquesta a escala global per tal de veure la manera de fer els protocols d'enregistrament i l'interès que l'aplicació crea en el sector. Aquest aspecte potser es va fer massa tard, ja que quan es va realitzar, el prototip estava bastant avançat. Si en canvi s'hagués fet en un primer moment, tot el disseny funcional hauria pogut estar basat en els resultats obtinguts. Tot i així, s'està molt satisfet de la rebuda que ha tingut i l'interès que ha generat l'enquesta realitzada. En ella s'ha pogut comprovar que el disseny anterior no estava molt desencaminat de la demanda del sector. No obstant això, cal remarcar la importància que té el *feedback* rebut i saber-lo aplicar al projecte.

Finalment, després de tot el procés d'implementació, s'ha pogut assolir l'objectiu inicial del treball en forma de prototip, tot i que a aquest encara li falten bastantes millores abans de poder sortir al mercat. Algunes són aspectes inclosos al disseny inicial però que donada la complexitat, s'han postposat a la segona fase d'implementació, d'altres són aspectes o idees que han anat sorgint a mesura que s'anava desenvolupant el projecte i d'altres són suggeriments que s'han rebut al qüestionari.

La usabilitat actual permet veure les possibilitats que té l'aplicació i engresca a seguir aquesta aventura intentant que arribi a bon port. Ara mateix però, només es podria fer algun enregistrament no massa complex. Vista la bona rebuda que ha tingut la notícia al sector i les mostres de suport rebudes al fer l'enquesta, hi ha la voluntat de tirar-lo endavant durant l'any vinent.

APÈNDIX A

ANNEX

A.1 Respostes de l'enquesta:

Mostra dels resultats obtinguts amb l'enquesta realitzada durant els mesos de Març i Abril del 2017 mitjançant GoogleForms[1]. L'última actualització és a dia 26 d'abril de 2017 tot i que a la data d'entrega, l'enquesta segueix oberta per poder recopilar més opinions de cara a futures actualitzacions de l'aplicació.

A.1.1 Sobre l'enquestat:

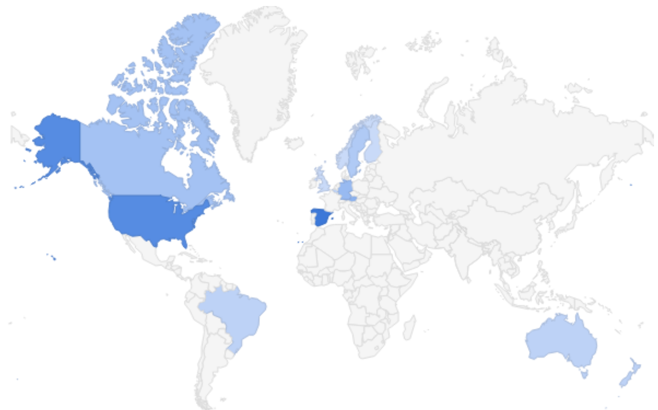


Figura A.1: Distribució al mapa dels països de procedència dels enquestats

[1] https://goo.gl/forms/Rn5OyilmNkionYLB3

Països de procedència:

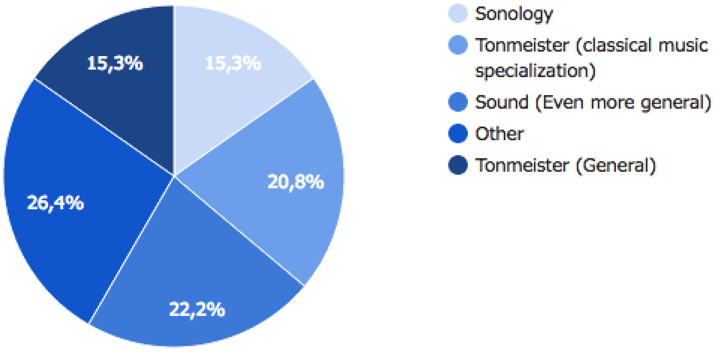
Anglaterra	17 respostes
Catalunya	15 respostes
Estats Units	12 respostes
Àustria	6 respostes
Alemanya	5 respostes
Canadà	4 respostes
Suècia	3 respostes
Austràlia	2 respostes
Bèlgica	2 respostes
Nova Zelanda	2 respostes
Brasil	2 respostes
Finlàndia	1 respostes
Noruega	1 respostes
Espanya	1 respostes

Universitats on han estudiat:

Escola Superior de Música de Catalunya	Catalunya	12 respostes
Surrey University	Anglaterra	10 respostes
Universität für Musik und darstellende Kunst Wien	Àustria	7 respostes
McGill University	Canadà	4 respostes
Northwestern University	Estats Units	2 respostes
Universitat Pompeu Fabra	Catalunya	2 respostes
Erich-Thienhaus-Institut (Detmold)	Alemanya	2 respostes
Royal Conservatory The Hague	Holanda	2 respostes
University of Southampton	Anglaterra	1 resposta
Royal College of Music	Anglaterra	1 resposta
Cambridge University	Anglaterra	1 resposta

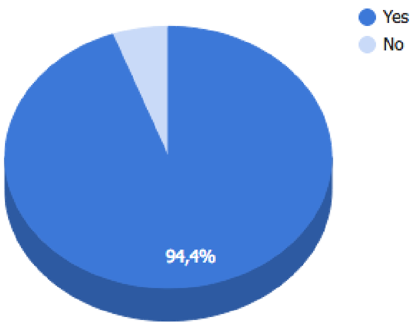
La Salle	Catalunya	1 resposta
Universitat Politècnica de Catalunya	Catalunya	1 resposta
Wesleyan University	Estats Units	1 resposta
University of Massachusetts Lowell	Estats Units	1 resposta
Ithaca College	Estats Units	1 resposta
Ohio University	Estats Units	1 resposta
Stony Brook University	Estats Units	1 resposta
Louisiana State University	Estats Units	1 resposta
Full Sail University	Estats Units	1 resposta
The Juilliard School	Estats Units	1 resposta
Manhattan School of Music	Estats Units	1 resposta
Universität der Künste Berlin	Alemanya	1 resposta
Musikhochschule Lübeck	Alemanya	1 resposta
Hochschule für Musik Hamburg	Alemanya	1 resposta
Linköping University	Suècia	1 resposta
Piteå School of Music	Suècia	1 resposta
Uppsala and Göteborgs University	Suècia	1 resposta
University of Adelaide	Austràlia	1 resposta
SAE Institute Melbourne	Austràlia	1 resposta
Institut des Arts de Diffusion	Bèlgica	1 resposta
Koninklijk Conservatorium Brussel	Bèlgica	1 resposta
Civica Scuola di Musica	Itàlia	1 resposta
Federal University of Pernambuco	Brasil	1 resposta
Scola Cantorum Basilensis	Suïssa	1 resposta
University of Edinburgh	Escòcia	1 resposta
Norwegian Music Academy	Noruega	1 resposta
Tornion taiteen ja viestinnän oppilaitos	Finlàndia	1 resposta
University of Auckland	Nova Zelanda	1 resposta

Estudis realitzats:



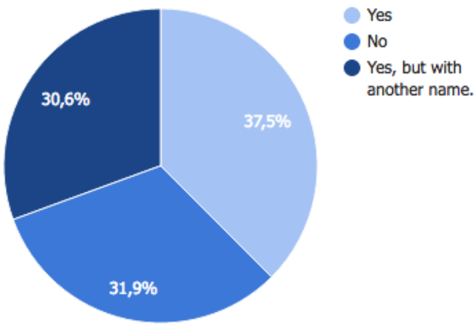
Altres: Orchestral Music, Psychology, Thetoric, Music, Performance, Telecomunications, Electrical engineering, Music Composition, Sound and Vibration Engineering (Acoustics), Audiovisual systems engineering, Economics i "Autodidactes".

Està treballant actualment de la música?



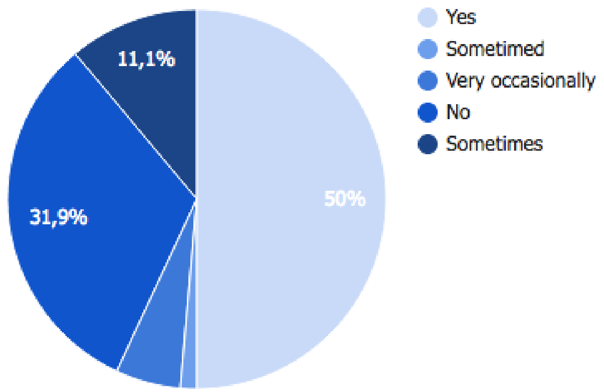
A.1.2 Sobre el protocol d'enregistrament:

Ha sentit a parlar sobre el protocol d'enregistrament?

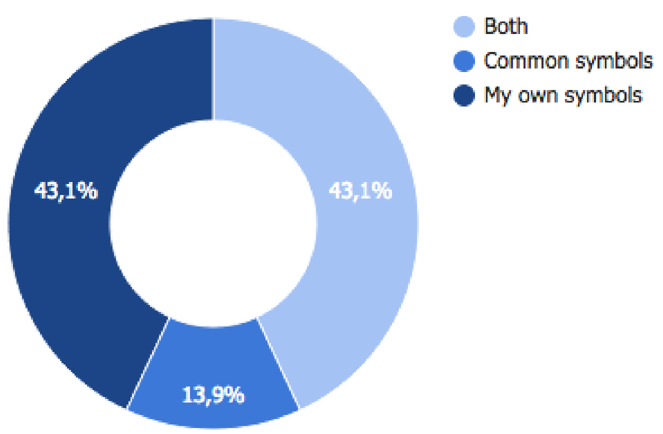


Altres noms: Take edit list, Score Marking, Take notes, Take sheet, Session logging, Score marking, Schnittplan, Partitureintragungen, Take protocol, Tape protocol, Redigeringsplan i Recording Log.

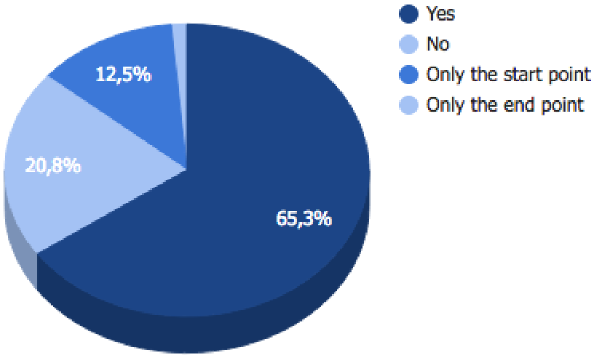
L'utilitza?



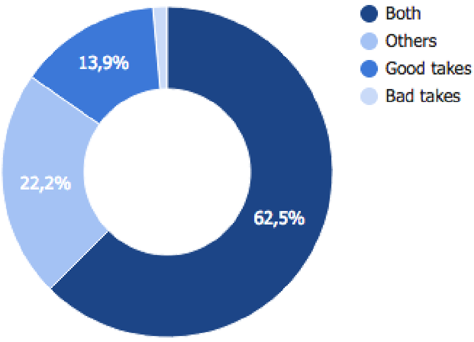
Utilitza símbols propis o d'apresos?



Marca l'inici i el final de cada presa?

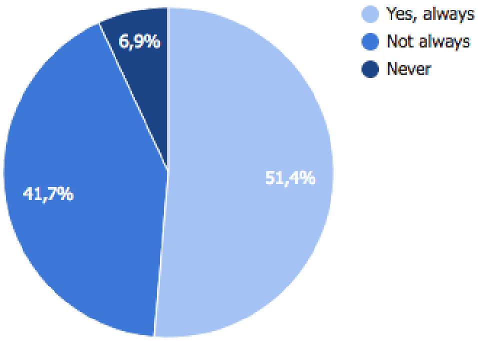


Que marca normalment?



Altres: Errors resoltos, canvis de tempo, entrades falses, sorolls, zones que s'han de resoldre i problemes varis (afincació, desajustos, etc...)

Normalment comença els enregistraments des de la presa 1?



En cas contrari, quin és el criteri?

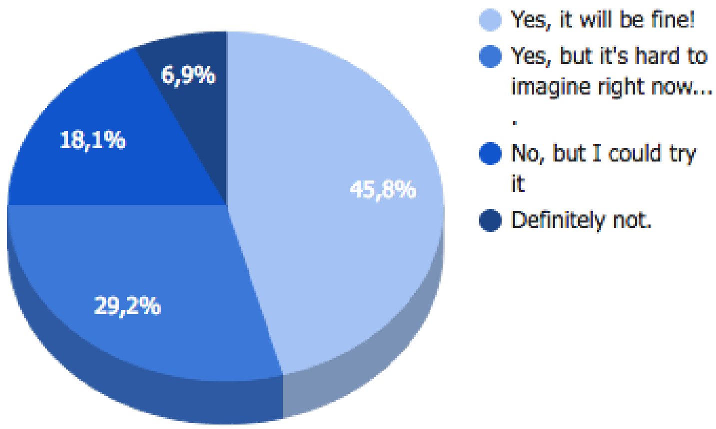
- The file number given from the machine.
- Multi day scoring session.
- The numbers often go one through the whole session, not every piece of music starts at one.
- Depends on situation, want to record comments between takes.
- Let's say we are recording in the morning and then again in the afternoon. I will start the morning first take with take 1. Then when we resume again in the afternoon, I continue with take number 121 or whatever it is. EVERY take has a unique number. Otherwise it is confusing.
- I may start counting from a previous take if a song has been previously recorded in the same session.
- The first take of the session is take one, takes count up from there regardless of the piece of music. This means that there are never two takes with the same number.
- Etiqueto les takes amb el nº de marker del Protools. Però a vegades hi ha markers que no corresponen a cap presa (Poden ser del tipus: sessió matí, sessió tarda, després de fer una pausa, etc). Jo tendeixo a anotar coses musicals i moltes d'altres extra-musicals: Han fet una pausa, estava tothom de molt mal humor, el cantant ha hagut d'atendre una trucada urgent i després ha seguit. És a dir coses que jo se que afectaran, o poden afectar, a l'interpretació i que em poden complicar els edits i per tant la presa de decisions sobre les takes bones.
- Because sometimes the first take (take one) it's just the test take, so then we start recording with the second take.
- En funció de si estic segur que el que tinc abans es pot aprofitar, començo la següent presa poc abans de l'error.

- There is no point in waisting recording time in renaming to take1 if the situation arises.
- Time stamp.
- It makes sense to start e.g. every movement with new numbers, especially if there will be a lot of takes. However, sometimes it can be easier to continue with the take numbers, and occasionally there simply is not enough time to make the proper adjustments in the recording machine.
- Sometimes an entire session is in one long file, especially if I want to hear the 'in between' talking and discussion.
- Keep advancing numbers.
- Just continually, throughout the session, not restarting the take count.
- I am recording different pieces in the same project and the takes just run through. The take number can
- Be checked in the take list. note: I don't know what çlassical recording protocolïs, and have never annotated scores while recording, so all my answers after Q1 are not valid.
- Some takes may be sound checks which makes the first recording take a higher number than one.
- In the old days DAT tape 1 starts with take 101 and ended if possible with 199. DAT tape 2 started with 201 etc. Today I use MOD and harddiscs, here I start with take 101 to 999 as a tradition
- Sometimes sequential take numbers over a whole project.
- Marking the beginning and end of each index in the score. I start again at 1 when I use a new hard drive.

- If the session was carrying on from another recording occasion several months previously, I would continue the numbers so there is only one T1 for a given album.
- I usually start a new movement / song / piece with 1 + a letter. This is to avoid having to write very long numbers as we go into the hundreds. So, the first piece would for instance have takes A1 - A15, the second B1 - B32 and so on. However, I would only make a note about the letter in the score at the beginning of the relevant piece / movement. All the subsequent marking would be numbers only. If we return to a movement / song later in the sessions, we pick up the letter and number relevant to that movement. This way there's never confusion about about which "take 32"relates to which movement. I also always run a separate "take log"on a separate sheet of paper, with every take logged with start point, and sometimes end point.
- I number takes throughout the session, over many days. Almost always use consecutive takes. Only restart from Take 1 when using separate take groups on discrete pieces in the recording on a recording likely to reach high numbers of takes.
- A unique take number for every take in a project.
- If TV/film is involved with time of day timecode then we may use another approach.
- If, by 'new recording', you mean the start of a new project, then 'yes'. If you mean by 'new recording', the start of a new piece of music or movement within a session, I keep the take numbers going to 'infinity'.
- I usually start a whole session from take one; the session might include several different pieces and we just keep going with consecutive numbers.
- Agreement with performers on which take to work from.
- If it's a separate piece it begins with take 1, if it's part of a larger piece or film I carry on starting with the next sequential take.
- Starting from the entry to a section which is being focused on.

A.1.3 Sobre l'aplicació:

Creu que podria prendre notes mitjançant l'ipad?



Que espera d'una apliació per a fer protocols d'enregistrament?

- Facilitar la feina del productor
- A complete editing plan
- Rapidesa, "netedat/ordre"els putucols a mà sempre acaben sent un picasso... molaria que les notes apuntades durant el fuckincabbage es guardessin com a markers al daw... una anada d'olla ja seria que t'ordenés les pistes per preferència (en playlist?) o et guardés les bones...
- Crec que una aplicació de protocol ha de ser molt intuïtiva, molt clara i senzilla visualment i donar una resposta molt ràpida a l'usuari. Hauria de ser més ràpid i senzill que fer les anotacions a llapis.
- To be very fluid
- That It gives an EDL (or similar) when the recording is done.
- Font Should be big enough. Clear layout
- Paperless work

- Quick notes, standard nomenclature, export to DAW
- Fast access to notes and guidance information, and easy to record this information
- I would expect to mark on an ipad version of the score my own symbols. I would need various colors to do it.
- Quick markup functionality, able to point good or bad interpretations in specific points of the score quickly.
- Ingest score in PDF; tool bar to select a variety of tools (mark start, end, misc. note, mark mistake, mark external noise); save and clear score markings after every take; would prefer touchscreen PC or Ipad pro for larger screen; make general take notes, like false start, timecode, musical notes like tempo, dynamics, etc. Hardware keyboard text entry very important for speed. Overview showing what sections are 'in the can' and what sections still need recording/retakes. Export to spreadsheet if possible.
- We've tried this before and there is still no faster means than a pencil and the score to make detailed notes.
- Organization!
- Que hi pugui escriure, dibuixar el que jo vulgui damunt l'score. En alguns casos (Sobretot en música contemporània) en l'actualitat he fet servir una aplicació per iPad que reprodueix partitures en pdf i en la que hi puc escriure, posar dibuixos, marques... Suposo que la coneixes. Es diu piaScore one must be able to use it very quickly
- Ease of use
- To improve the classical approach of writing down directly to the score. Should be fast, reliable, interchangeable, edit-friendly. Would be great if it could synced with the acual recorded take and exported into a DAW as timecoded marks.

- Speed (must be faster than a pencil on paper), export to desktop to make all my usual score- marks in this application. Not only start and end point of the take, good and bad take e.g., but also to be able to write all my other marks in the score. I write a lot of marks directly in the score: like -2 I"(= take 2 bad Intonation in this part) or - 3 B"for Balance problem in Take 3 in this specific part. And a lot more. Plus special marks for edit points, extra marks for noise, specific marks for musician comments etc. I use different color pencils for this too… You know this methods…. We should be able to customize this marks for ourselves too...Would be handy to do this in a very fast way on the iPad so we can leave the printed score at home. Right now i can't imagine to this without pencil and score. As you know, we have to to this marks while recording, so it has to be very fast and convenient. This would be the mandatory for me! Some more ideas: we should be able to scan in the scores by ourselves and use it. Is this planned? And there should be some kind of navigation possibilities, e.g. to be able to jump to a certain page, or bar, or mark or whatever. That would be handy. I think i have a lot more ideas- please tell me if i can help you with this app. Would be glad to have something like this for my work. And if it works really well, i would pay good money ;-) All the best for your development!

- Agility, clear and easy to work with it.

- Easier than colored pens.

- Synchronization between the score and the recordings.

- To make faster and easier to elaborate the recording protocol.

- To be able to export eg. a .pdf that can be used on another platform or for a printout.

- As easy to use a pencil on paper.

- I am so used to working with my hands, very fast, on a physical score, that an application would have to yield the feel of being ïn touch"with the score - so, it must be responsive and fast and allow me to directly make notes in between the notes etc. If I have to drag icons or select something, it will take too much time and focus.

- -Compatibility with Android and/or Windows. -Handwriting option. -Layers.
- It should allow to upload, save, and export. The files should be pdf or jpeg.
- I would want to be able to write directly on the score, zoom in to annotate, zoom out to score follow. You should be able to choose different colors for text, make highlighter-like markings (colored but see-through)
- I would love an app like this.
- I don't use iPads/Apple products so irrelevant to me. I have used laptop based session logging software for over 20 years (sometimes integrated with recording machine control software) and see no reason why it shouldn't work well on a tablet. I wish you good luck with your work and your degree.
- Simple to use.
- Easy to use.
- That it's connected to the DAW, providing me with a shortcut to mark/colour takes according to its quality and also if required, delete stuff right away.
- Not to crash.
- Let's talk, I wanted to develop one of my one, but never did.
- Quick, smooth score scrolling, quick notes with either a tap or drawing the note. (Also an Android version!!!).
- Pen-usable, quick in score-shifting, print in pdf.
- Fast(er than paper), single note accuracy, fast and accessible symbols for marking, good page flow every note you make should be saved in a easy-accesable-list with the information in which bar it happened and what kind of mistake. so after every

take you can go easily through the list, and check which corrections you have to do, and if there are correct versions of every part of the piece of to see if you forgot something.also it should recognise letters as a type of mistake. e.g. if you write an ï"on the viola part, in the mistake-list should appear "bar 47 - viola- intonation - note g"'. or if you draw a wavy line in the score in the list should apprear "bar 27 - instruments not together".

- Easy to write on the score.
- Ability to easily notate: the best takes for each section; possible inserts; sections that are bad that need to be replaced.
- That it is easy to understand and easy to handle during a recording.
- A good scrollable version of the score. a clear way to denote keeper takes. ease of use and speed
- Very fast to write and erase, wide variety of color, some common stamp automatic incrementing of take number.
- Clever use of press/long press/hard press different functions (hard press for marking a noise, long press for marking a good passage at cetera), good score handling including PDF imports into library, cataloging a set of scores into a folder for a specific recording session (leaving the score PDF untouched).
- Clear view of the score, ability to note in different colours, easy to use customisable symbols to denote various things, automated take number logging, ideally timecode input to link recording timecode with the notes
- I have no idea to be honest.
- To show different color areas in the score depending on the music problems or good options (in terms of takes) in the recording process.

- It would be good if you can overlay the score with each 'take' separately, or together as required and then have a menu to easily view them. So, for example, you could just see take 6, or perhaps take 3 and 6 over-layered to compare. After that, If there were common icons we could use to quickly markup, for example, a circle to drop on a good bit, or a cross to put over a wrong-note. Perhaps going into even more detail, we could mark what the problem was - so an icon for tuning, or ensemble etc. We could quickly and easily make notes on the take.

- Not sure at the moment.

- Can't currently turn a page and quickly switch back to red or green annotations.

- That the user would be able to write on the screen as if writing on the paper, with a pen and then the notes would show up on the score as if they were already there. Assuming that you're taking notes as the music is being recorded, it's definitely quicker to write the notes down than typing them on a keyboard. There could also be symbols which would be displayed on screen and the user would only have to drag them on top of the notes. Say, a symbol regarding tuning to mark a certain note as out of tune.

- Colors, easy notation of false start, easy region definition and assertion based on the Pyramix region markers(bad, ok, good, very good, amazing).

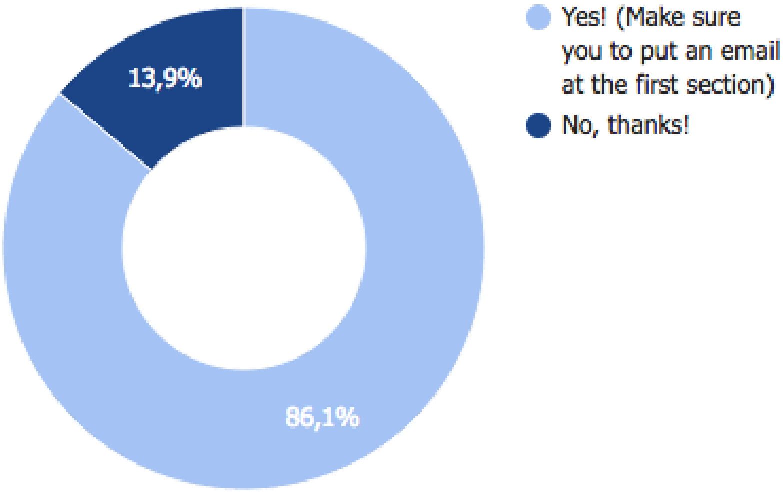
- fast to note things, no typing, use a touch screen for writing, and ability to erase mistaken notes very quickly.

- It would have to be extremely responsive - no delay at all. It would have to be very easy to turn pages.

- It would have to be only sensitive to a pencil-type input - normal finger touching must somehow be turned off so you don't accidentally mark the score while pointing or counting bars for example. It would have to be very easy to upload pages to it.

- I have never thought about this! My markings are very particular, and I don't know how to convey those to the rather more clumsy interface of an iPad.
- It has to be quick and easy .
- Intuitive and fast.
- Speed of use and response of writing. ability to get round score v quickly (ie turning pages, seeing whole score etc.
- As easy if not easier than doing it with a pencil, speed, ease of use, clarity, reliability
- Easy and quick to use and clear.
- Ability to use a pen, not type. Easy to flip through score pages, even search for a take number.
- I prefer the solidity of paper and pen.
- Everything that I can write in a paper score. Use of iPencil / iPen (whatever it's called)
- Drop down menus to enter quick take notes.
- It would have to be easier than my normal note taking.
- Simple ways to advance through takes quickly. A way to move back to previous takes and add or remove notes as needed. Support for text documents for narrations and voiceover would be wonderful. way to add any notes that could be added via paper score.
- easy to quickly apply symbols to specific bars/beats. It would be useful to be able to lad a score PDF in it and to be able to apply symbols to the score. Different coloured symbols for different takes?
- Time code, measure numbers or Letter symbols which correlates to the score.

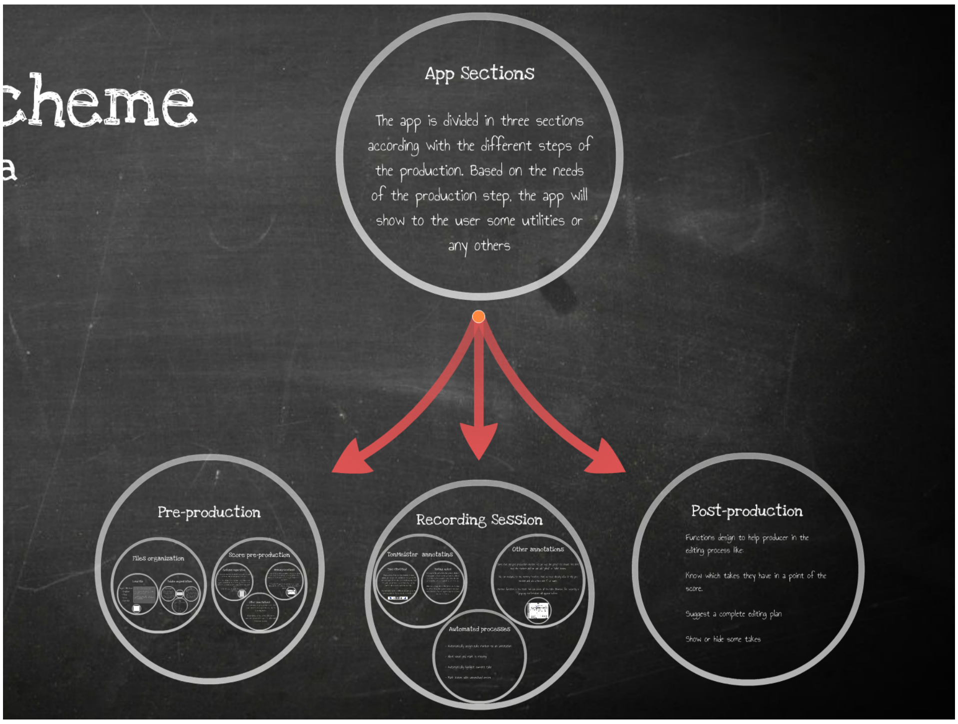
Vol estar informat quan l'aplicació surti al mercat?

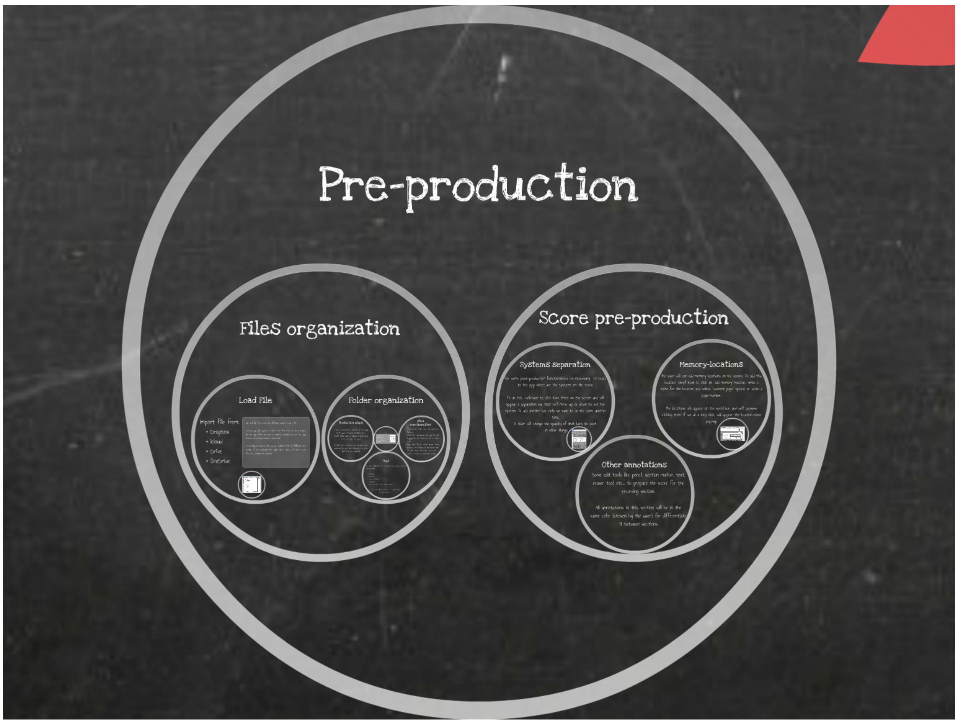


Té alguna referència que parli sobre el protocol d'enregistrament?

- I do not have specific references, but I feel this is a very interesting topic, and one that can be highly subjective. However, I have always thought that it makes a real difference if you are able to work efficiently with standard symbols, especially if other people will work with them, too (e.g. an editor). Therefore, it would be a good idea to standardise it. You may contact me if you want more info.
- `http://mark-rogers.com/index.php/software/lumberjack`
- `https://www.spottingnotes.com/frontpage.php`
- Recording Orchestra and Other Classical Music Ensembles by Richard King, Chapter 20. (this has a useful basic guide to the conventions of marking a score)

A.2 Esquema prezi

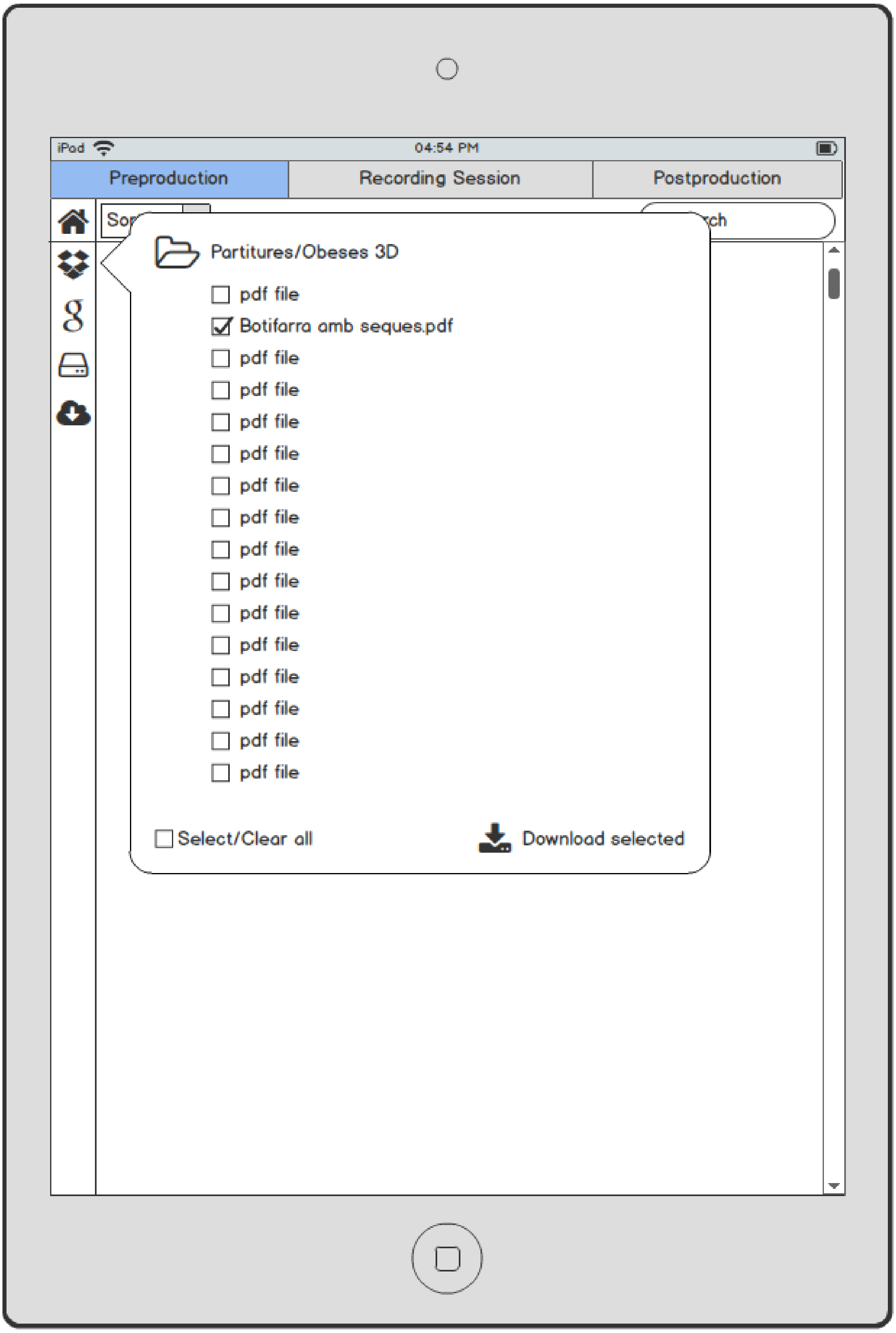


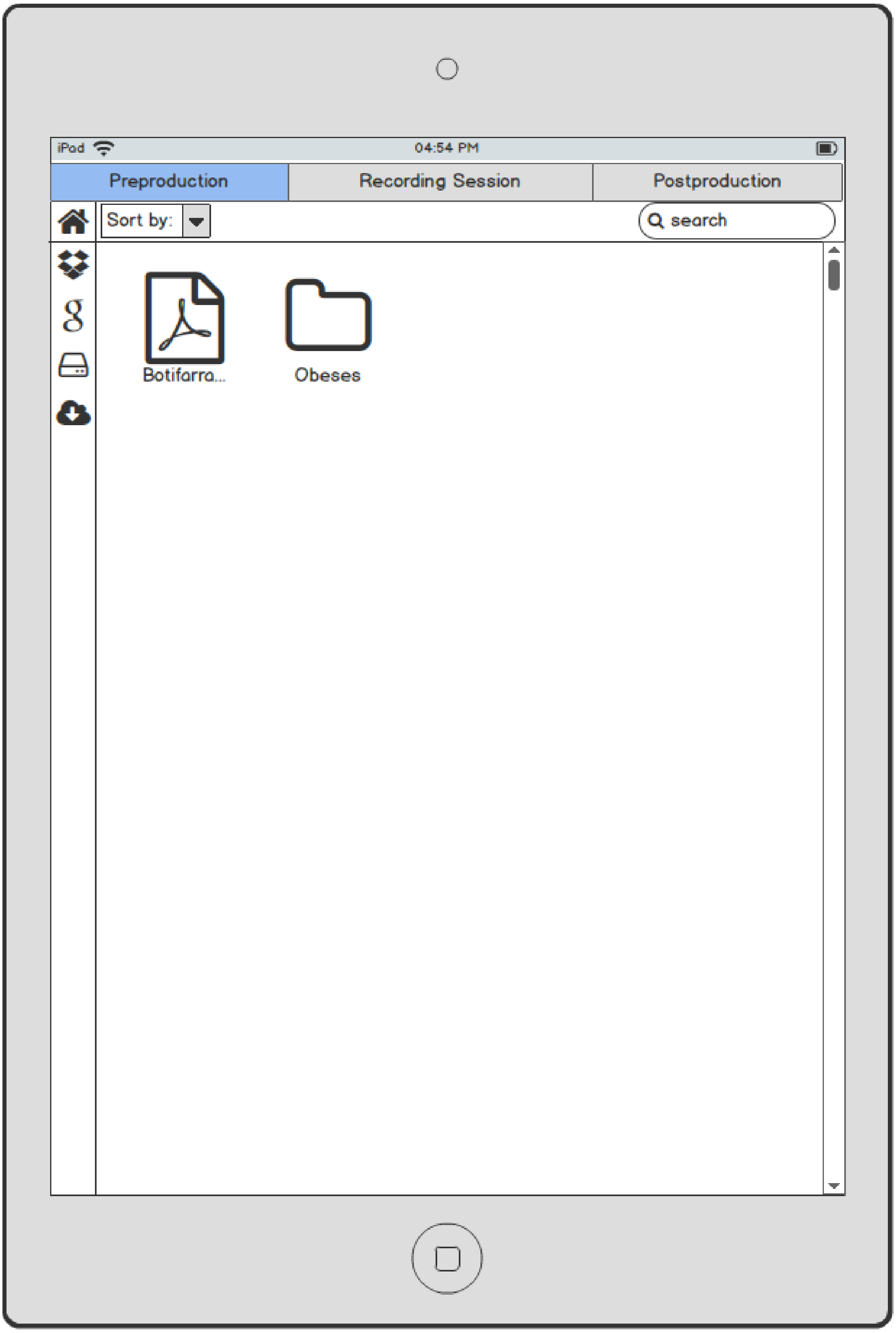


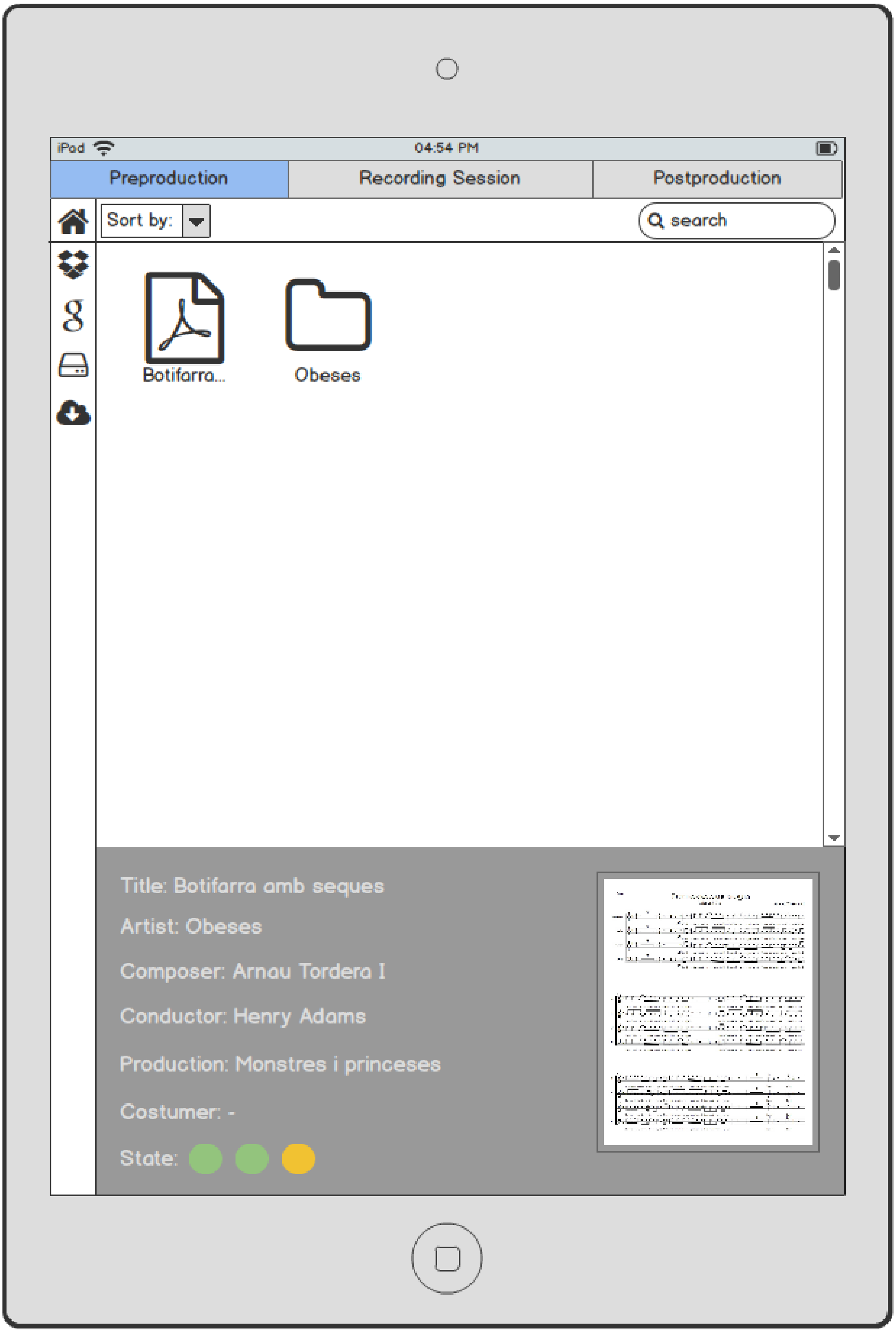

Recording Session
TonMeister annotatins
Take Start/End
Rating notes
Other annotations
Same that the pre-production section, we can use the pencil, the eraser, the text tool, the marker and we can add "global" or "take" notes.
We can navigate to the memory locations that we have already added in the pre section and add a new one if we want.
Another function is the trash. We can delete all the take. However, for security, a pop-up confirmation will appear before.
Automated processes
- Automatically assign take number to an annotation
- Alert when end mark is missing
- Automatically highlight current take
- Mark zones with unresolved errors

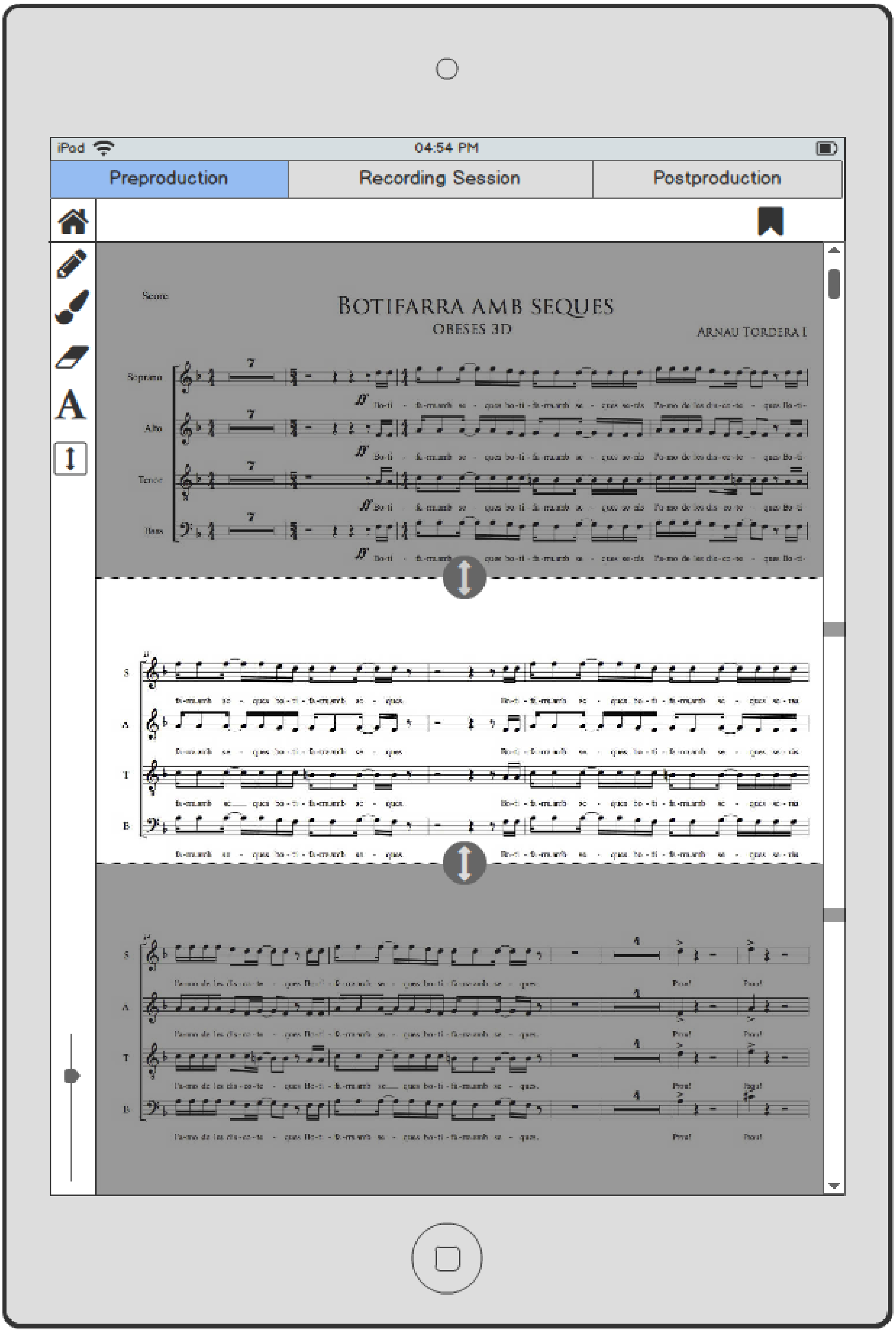

Post-production
Functions design to help producer in the editing process like:
Know which takes they have in a point of the score.
Suggest a complete editing plan
Show or hide some takes

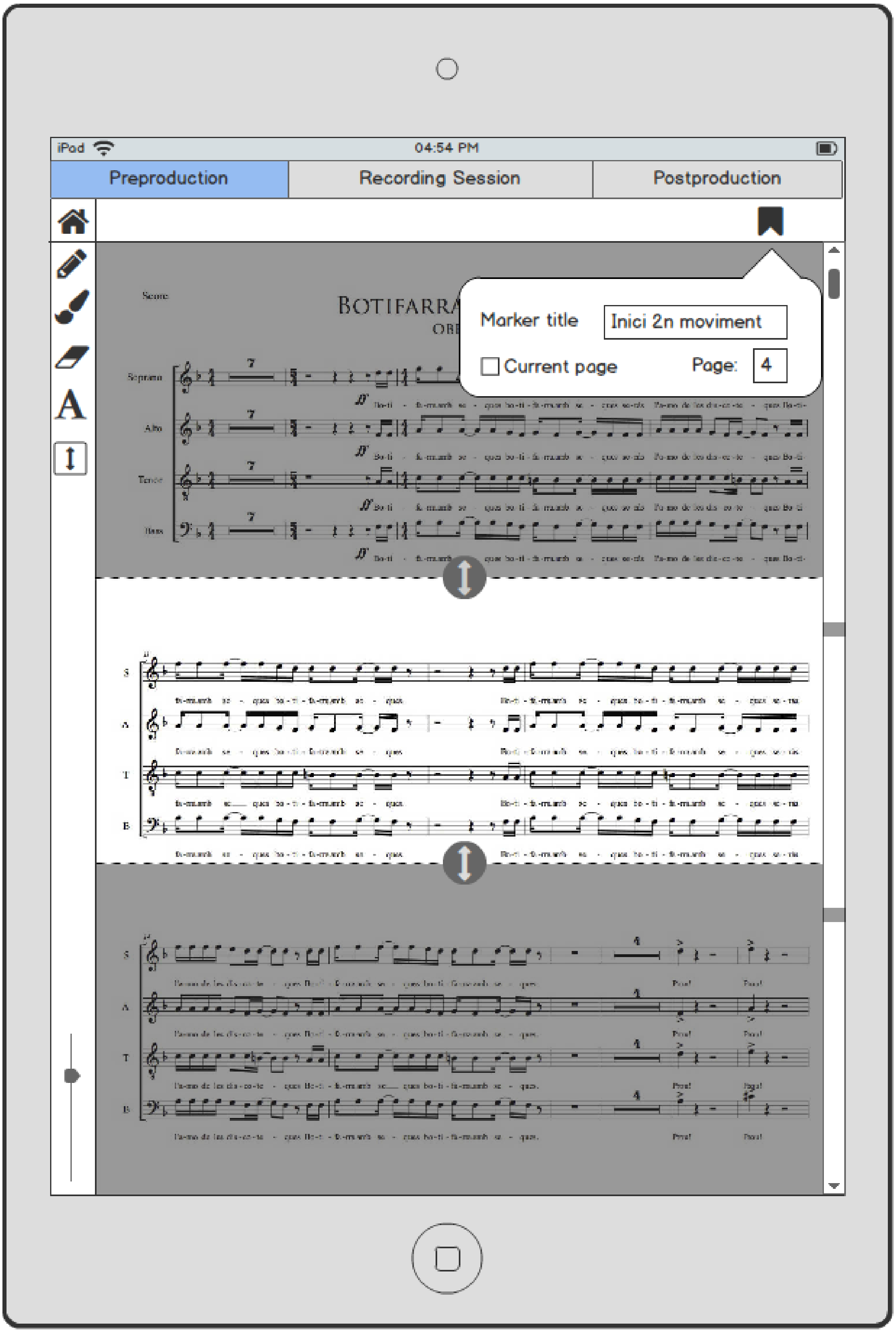
A.3 MockUp

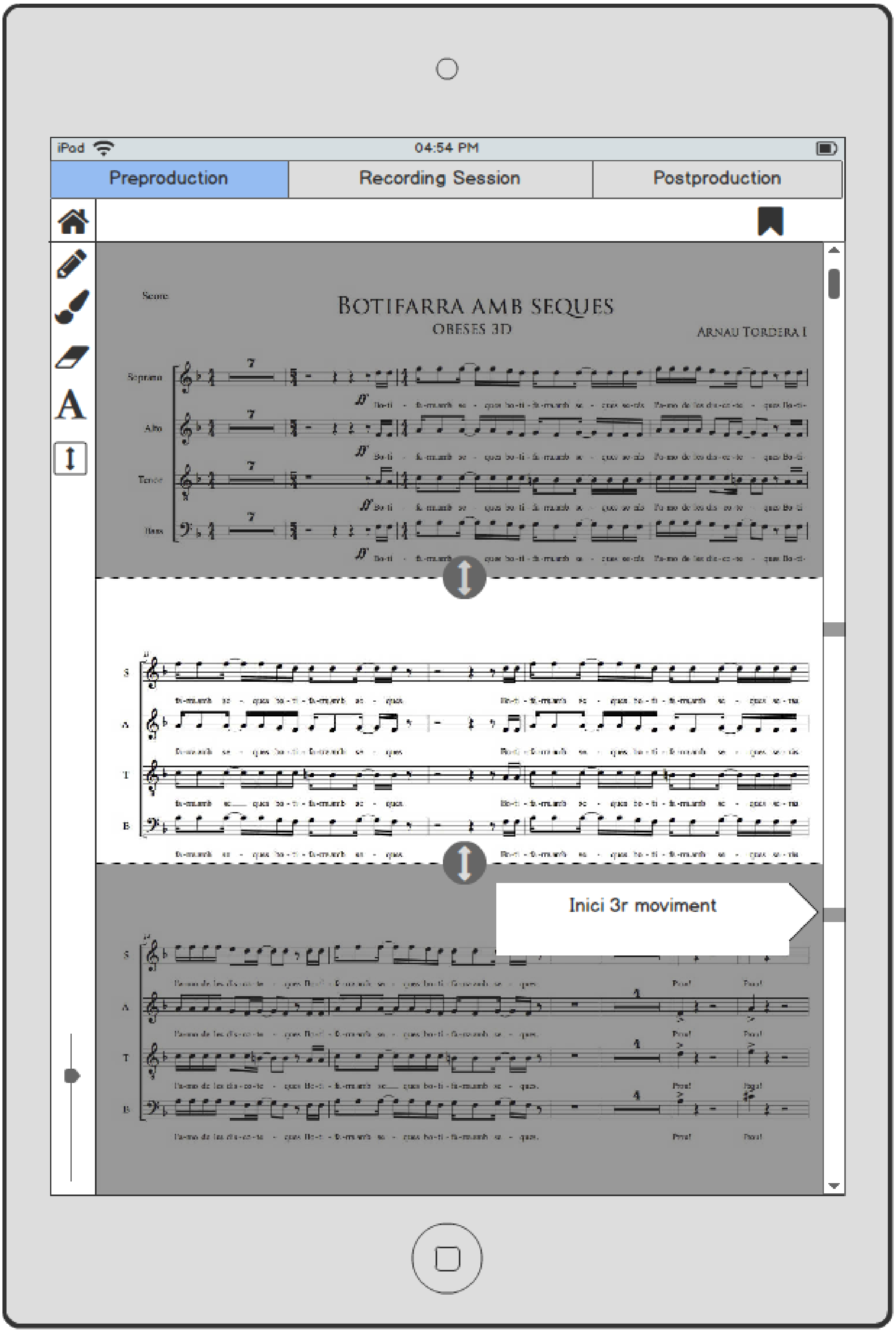

iPad
04:54 PM
Preproduction
Recording Session
Postproduction
Partitures/Obeses 3D
pdf file
Botifarra amb seques.pdf
pdf file
pdf file
pdf file
pdf file
pdf file
pdf file
pdf file
pdf file
pdf file
pdf file
pdf file
pdf file
pdf file
pdf file
Select/Clear all
Download selected

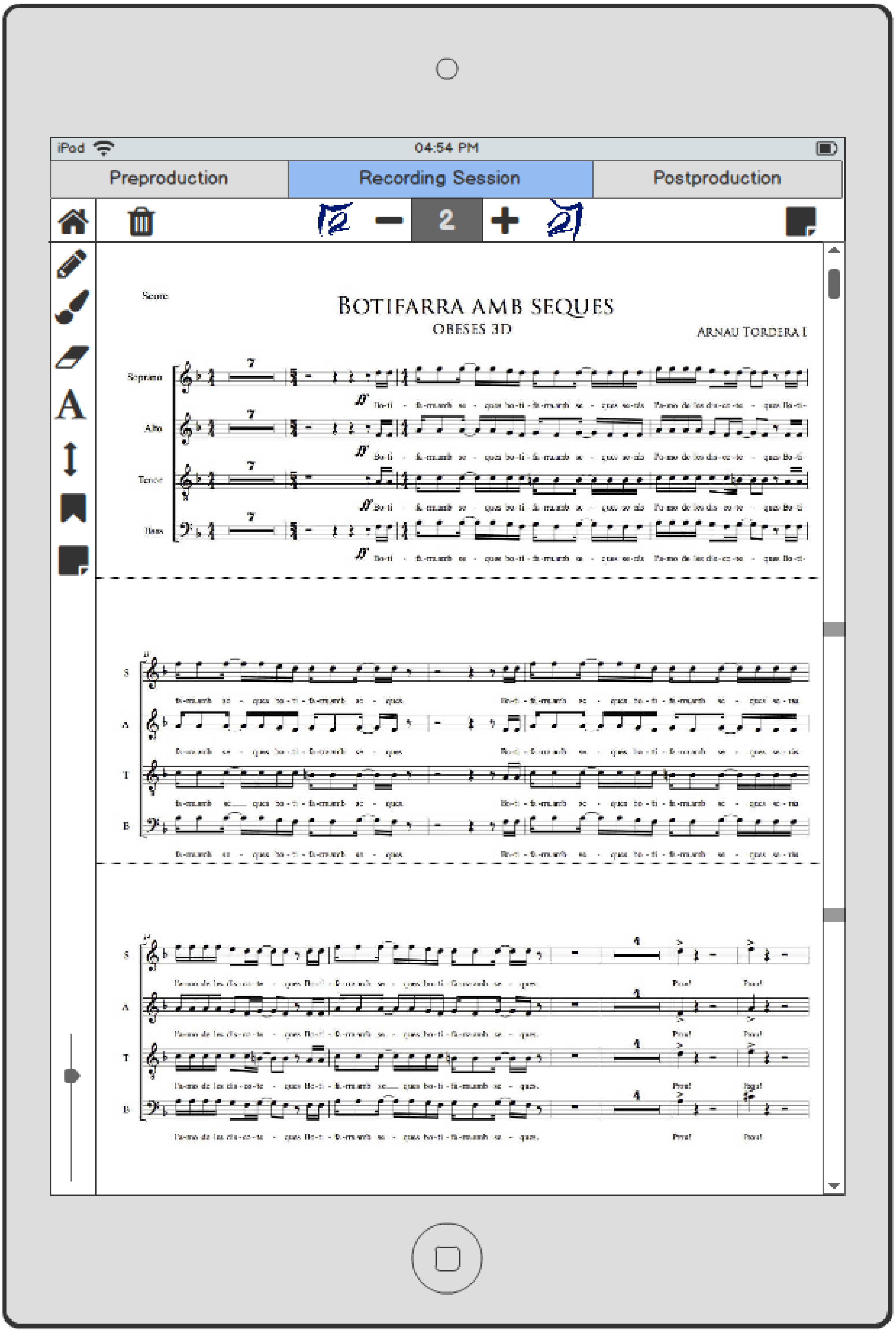

iPad
04:54 PM
Preproduction
Recording Session
Postproduction
Sort by:
search
Botifarra...
Obeses

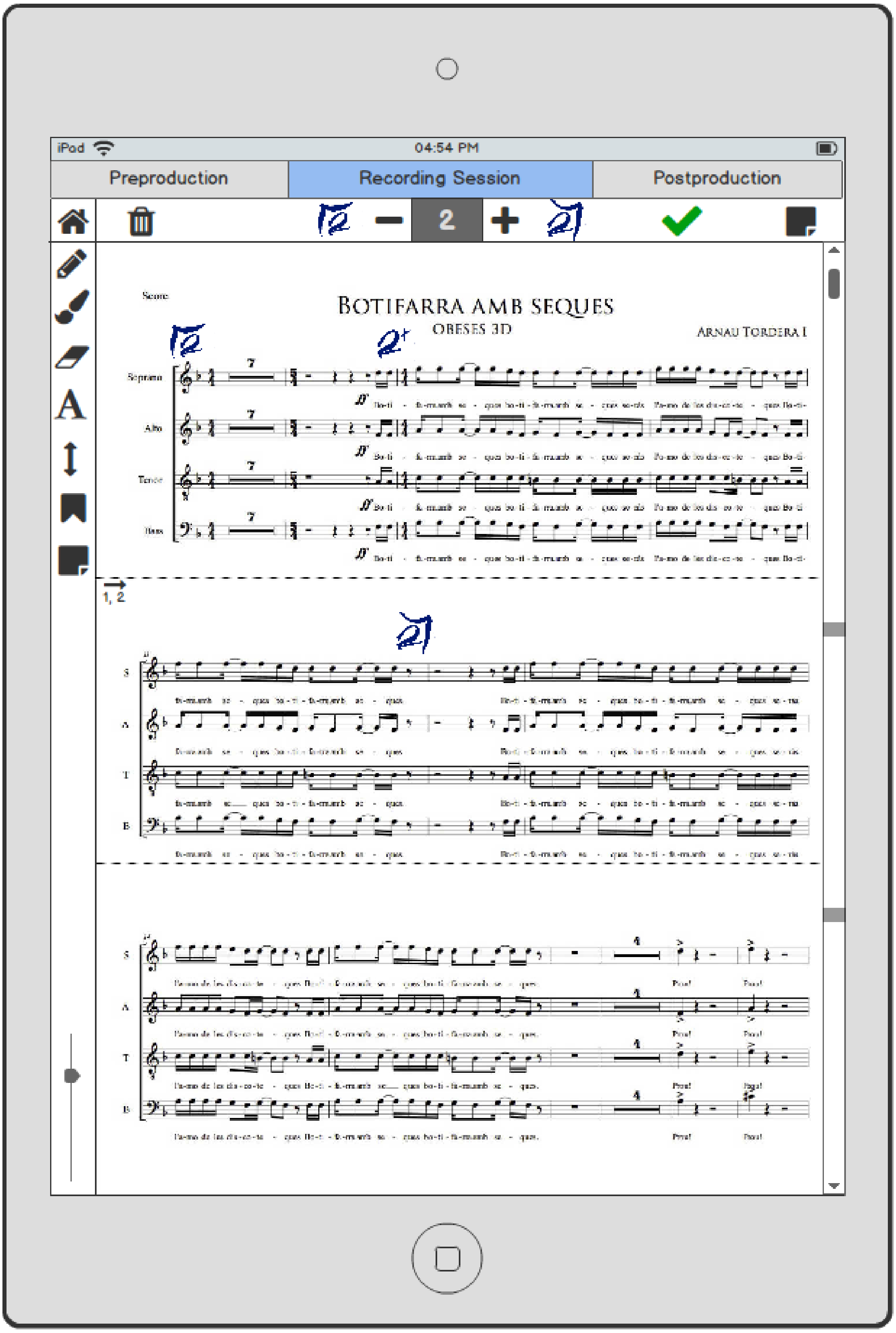

iPad
04:54 PM
Preproduction
Recording Session
Postproduction
Sort by:
search
Botifarra...
Obeses
Title: Botifarra amb seques
Artist: Obeses
Composer: Arnau Tordera I
Conductor: Henry Adams
Production: Monstres i princeses
Costumer: -
State:

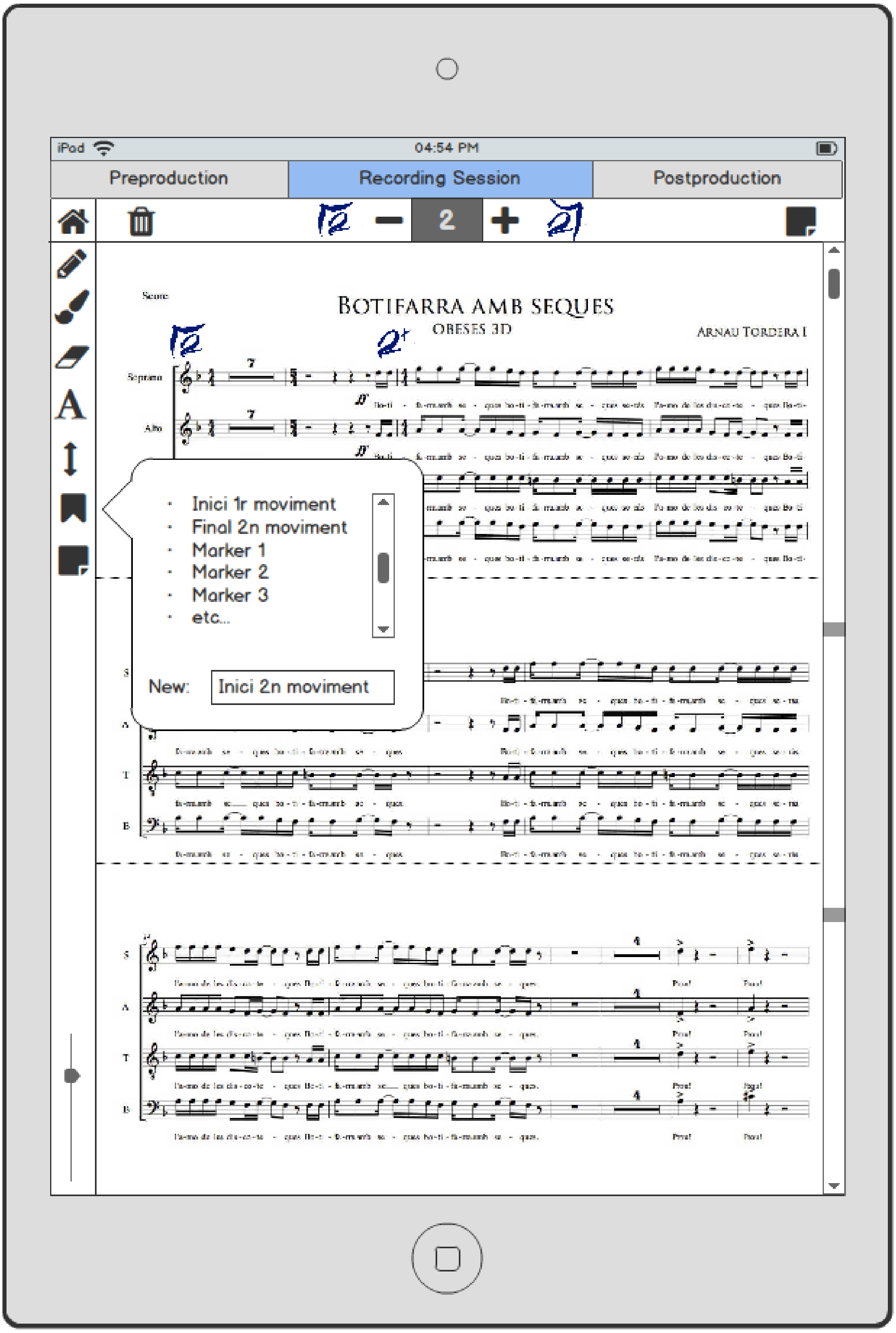

iPad
04:54 PM
Preproduction
Recording Session
Postproduction
Score
BOTIFARRA AMB SEQUES
OBESES 3D
ARNAU TORDERA I

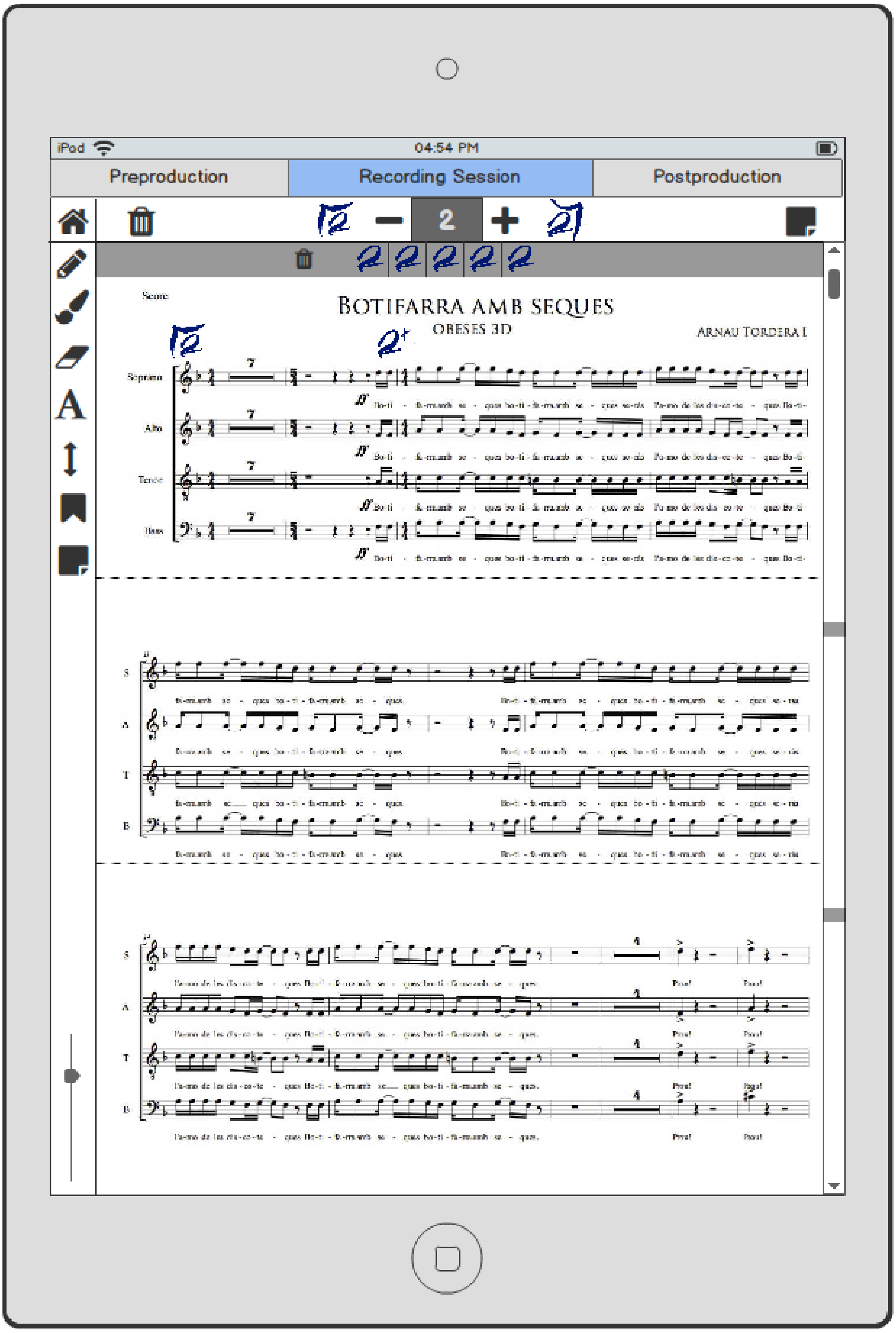

iPad
04:54 PM
Preproduction
Recording Session
Postproduction
Marker title
Inici 2n moviment
Current page
Page:
4
BOTIFARRA

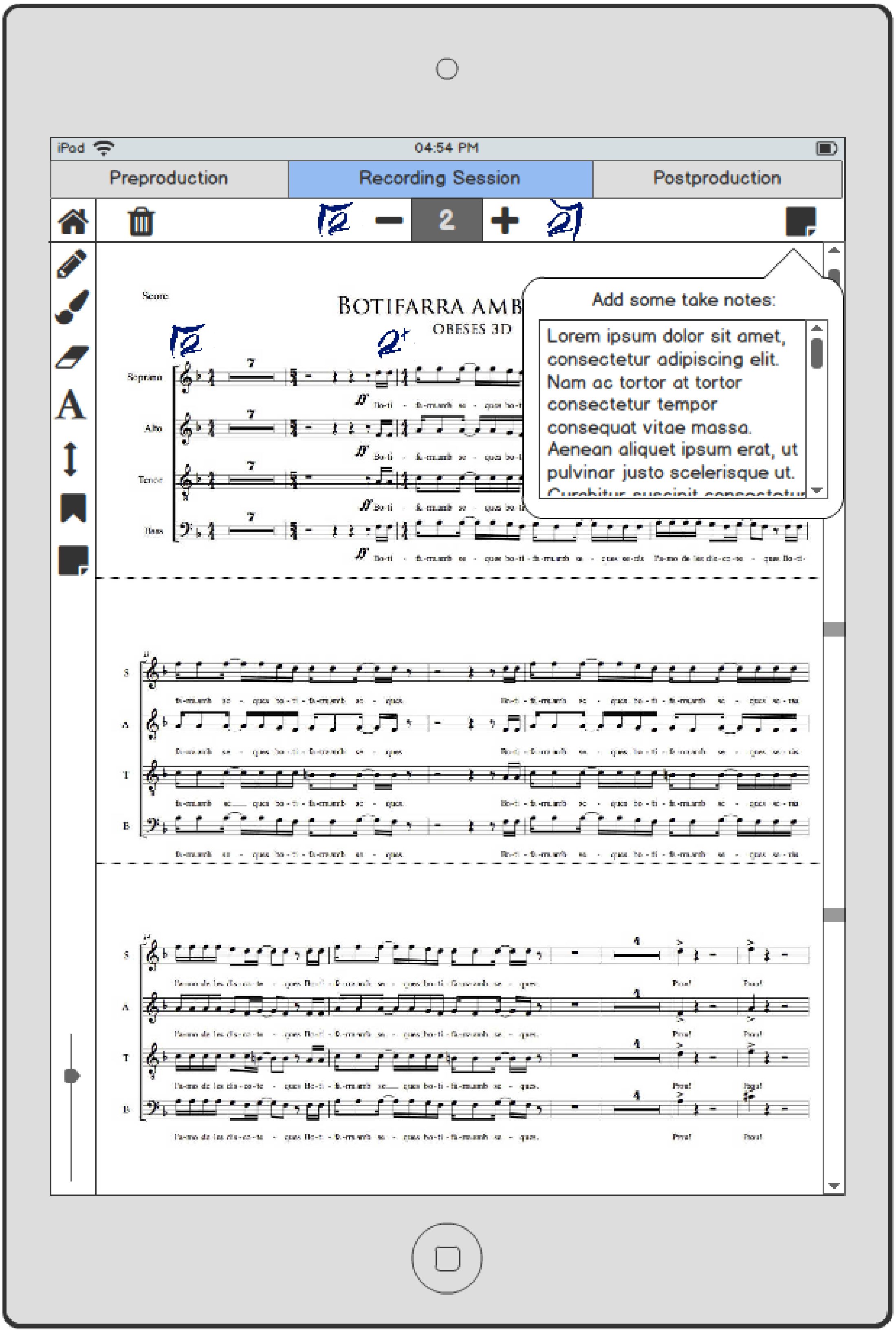

iPad
04:54 PM
Preproduction
Recording Session
Postproduction
Score
BOTIFARRA AMB SEQUES
OBESES 3D
ARNAU TORDERA I
Soprano
Alto
Tenor
Bass
Inici 3r moviment

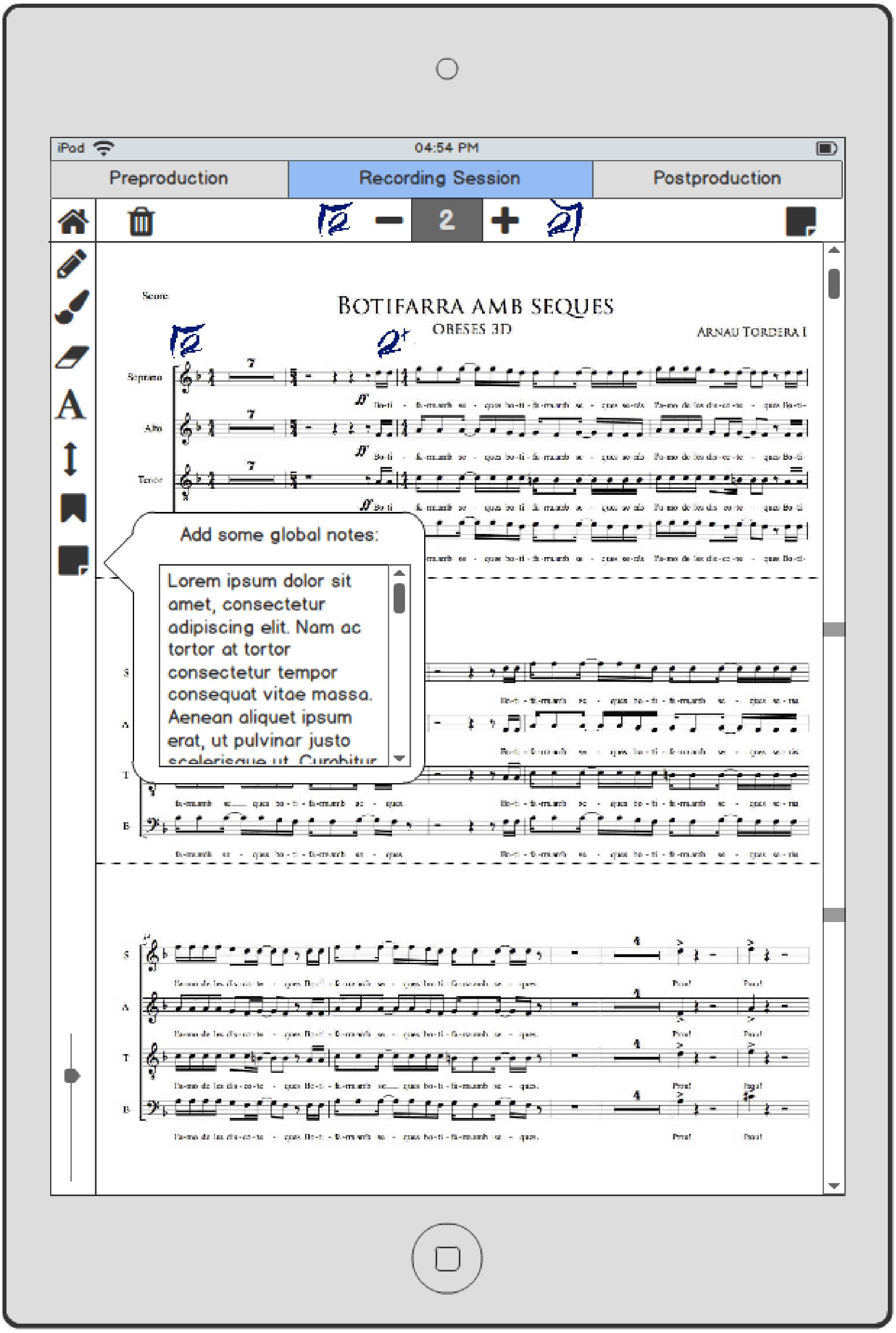

iPad
04:54 PM
Preproduction
Recording Session
Postproduction
2
BOTIFARRA AMB SEQUES
OBESES 3D
ARNAU TORDERA I

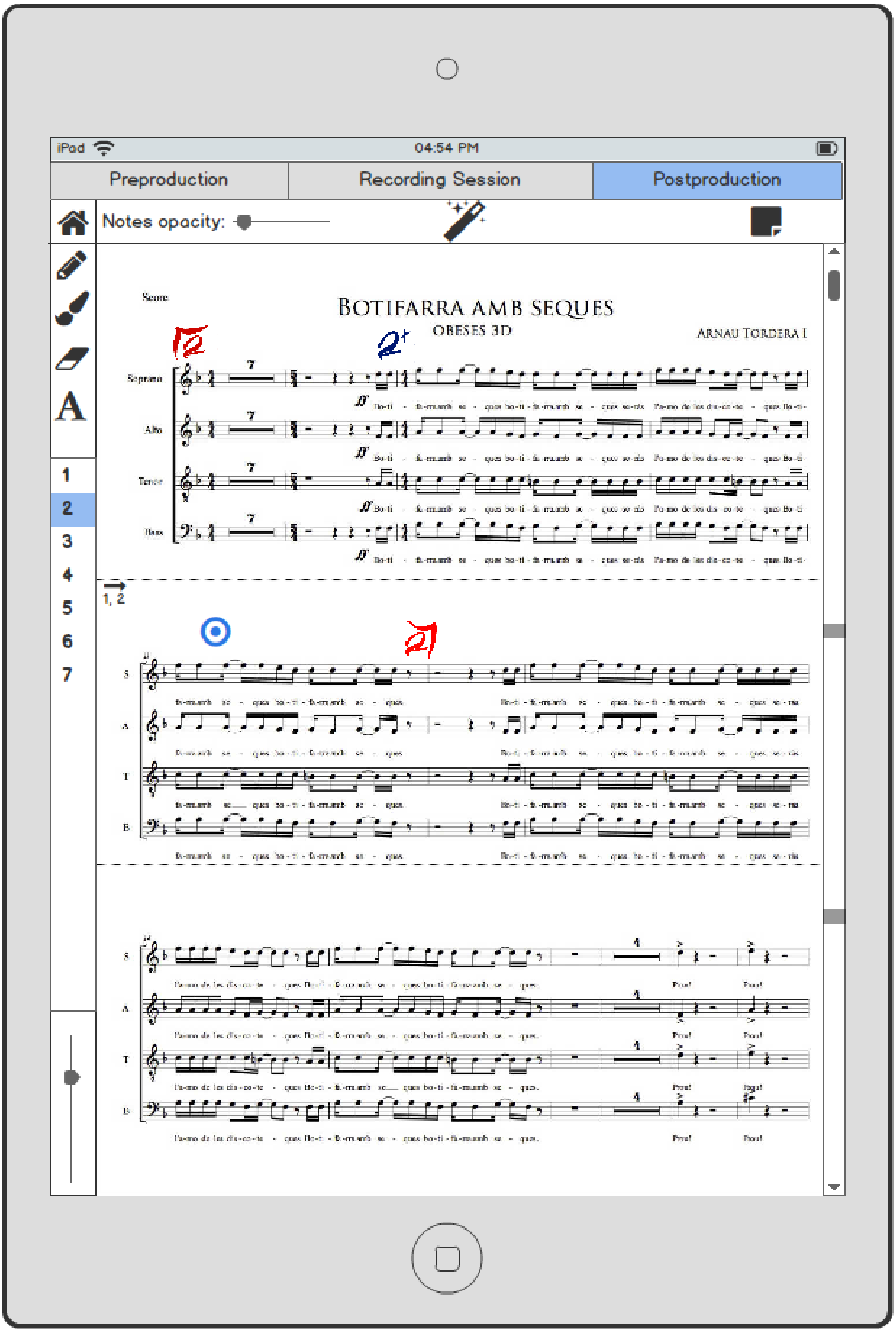

iPod
04:54 PM
Preproduction
Recording Session
Postproduction
2
Score
BOTIFARRA AMB SEQUES
OBESES 3D
ARNAU TORDERA I
1, 2

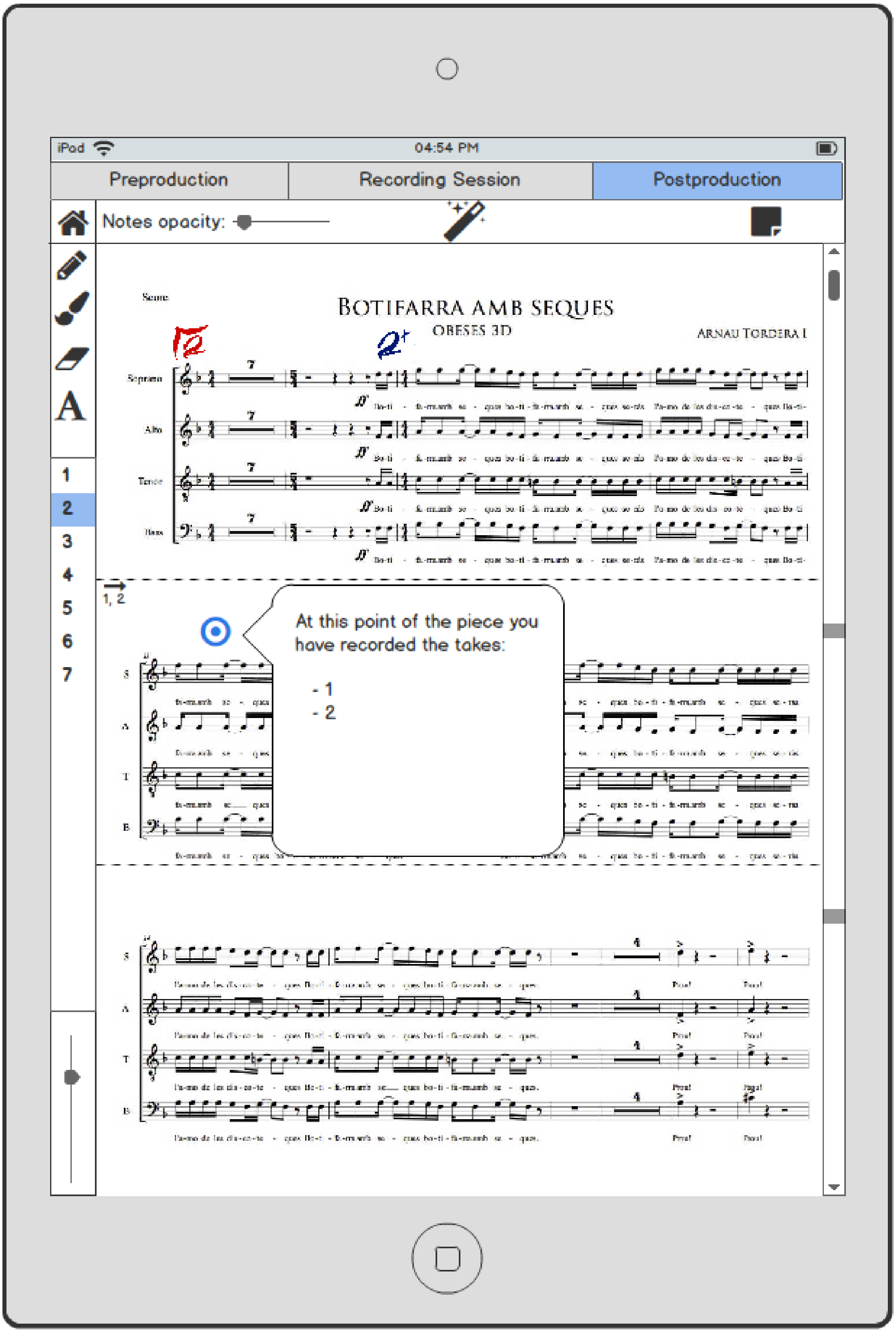

iPad
04:54 PM
Preproduction
Recording Session
Postproduction
2
Score
BOTIFARRA AMB SEQUES
OBESES 3D
ARNAU TORDERA I
Soprano
Alto
Inici 1r moviment
Final 2n moviment
Marker 1
Marker 2
Marker 3
etc...
New:
Inici 2n moviment


iPod
04:54 PM
Preproduction
Recording Session
Postproduction
2
Score
BOTIFARRA AMB SEQUES
OBESES 3D
ARNAU TORDERA I
Soprano
Alto
Tenor
Bass


iPad
04:54 PM
Preproduction
Recording Session
Postproduction
2
Score
BOTIFARRA AMB
OBESES 3D
Soprano
Alto
Tenor
Bass
Add some take notes:
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam ac tortor at tortor consectetur tempor consequat vitae massa. Aenean aliquet ipsum erat, ut pulvinar justo scelerisque ut.


iPad
04:54 PM
Preproduction
Recording Session
Postproduction
2
Score
BOTIFARRA AMB SEQUES
OBESES 3D
ARNAU TORDERA I
Soprano
Alto
Tenor
Add some global notes:
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam ac tortor at tortor consectetur tempor consequat vitae massa. Aenean aliquet ipsum erat, ut pulvinar justo


iPad
04:54 PM
Preproduction
Recording Session
Postproduction
Notes opacity:
A
1
2
3
4
5
6
7
Score
BOTIFARRA AMB SEQUES
OBESES 3D
ARNAU TORDERA I
Soprano
Alto
Bass
1, 2


iPad
04:54 PM
Preproduction
Recording Session
Postproduction
Notes opacity:
Score
BOTIFARRA AMB SEQUES
OBESES 3D
ARNAU TORDERA I
Soprano
Alto
Tenor
Bass
1
2
3
4
5
6
7
1, 2
At this point of the piece you have recorded the takes:
- 1
- 2
S
A
T
B

A.4 Codi

A.4.1 Secció Home

```
//
//  ViewController.swift
//  TFG-importFile
//
//  Created by Adrià Martínez on 31/10/15.
//  Copyright © 2015 Adrià Martínez. All rights reserved.
//

import UIKit
import SwiftyDropbox
import CoreData
import ADAL
import M13PDFKit


class HOME_ViewController: UIViewController, UITableViewDelegate,
    UITableViewDataSource
{

    // Declaració dels objectes de la UI
    @IBOutlet var btnPre: UIButton!
    @IBOutlet var btnRec: UIButton!
    @IBOutlet var btnPost: UIButton!
    @IBOutlet var btnTrash: UIButton!
    @IBOutlet var btnDone: UIButton!
    @IBOutlet var btnLogOut: UIButton!
    @IBOutlet var btnLogIn: UIButton!
    @IBOutlet var btnEditClear: UIButton!
    @IBOutlet var btnEditSelectAll: UIButton!

    @IBOutlet var taulageneral: UITableView!

    var editSel = [String]()


    override func viewDidLoad()
    {
        super.viewDidLoad()

        HomeSeccActiva()
        NotificationCenter.default.addObserver(self, selector: #selector
            (HOME_ViewController.reloadTableData(_:)), name: NSNotification.Name
            (rawValue: "reload"), object: nil) //Observer per fer refresh
        self.reload()

        let clickLlarg = UILongPressGestureRecognizer(target: self, action:
            #selector(HOME_ViewController.editTable(_:)))
        clickLlarg.minimumPressDuration = 1

        self.taulageneral.addGestureRecognizer(clickLlarg)

        self.btnDone.isHidden=true
        self.btnTrash.isHidden=true
        self.btnEditSelectAll.isHidden=true
        self.btnEditClear.isHidden=true
        sistemesON=false
    }
```

```
////// BOTONS ///////

@IBAction func pre(_ sender: AnyObject)
{
    seccioActiva=1

}


@IBAction func rec(_ sender: AnyObject)
{
    seccioActiva=2
}


@IBAction func post(_ sender: AnyObject)
{
    seccioActiva=3
}

@IBAction func logInDrop(_ sender: AnyObject) //Funció per loggejar-se a
    Dropbox
{
    if (DropboxClientsManager.authorizedClient == nil)
    {
        //print(DropboxClientsManager.authorizedClient)
        DropboxClientsManager.authorizeFromController(UIApplication.shared,
                                                      controller: self,
                                                      openURL: { (url: URL) ->
                                                     Void in
                                                        UIApplication.shared.
                                                     openURL(url)
        })
        //print(DropboxClientsManager.authorizedClient)
    }
    else
    {
       // print ("ja estas logguejat amb aquest codi:")
       // print(DropboxClientsManager.authorizedClient)
    }


}

@IBAction func logOutDrop(_ sender: AnyObject) //I per sortir
{
    DropboxClientsManager.unlinkClients()
}


@IBAction func editDone(_ sender: AnyObject)
{
    taulageneral.isEditing=false
    btnTrash.isHidden=true
    btnDone.isHidden=true
    btnLogIn.isHidden=false
    btnLogOut.isHidden=false
    btnEditClear.isHidden=true
    btnEditSelectAll.isHidden=true
}
```

```
@IBAction func removeMultipleFiles(_ sender: AnyObject)
{
    let alert = UIAlertController(title: "Delete multiple files", message:
        "Are you sure you want to delete all files?", preferredStyle:
        UIAlertControllerStyle.alert)
    alert.addAction(UIAlertAction(title: "Yes", style: .default, handler: {
        (action) -> Void in

        for i in 0...self.editSel.count-1
        {
            self.deletePDFfromCoreData(self.editSel[i])
        }
        self.editSel.removeAll()
        self.reload()
        self.btnTrash.isHidden=true

    }))

    alert.addAction(UIAlertAction(title: "No", style: .default, handler: {
        (action) -> Void in

    }))

    self.present(alert, animated: true, completion: nil)


}


@IBAction func editSelectAll(_ sender: AnyObject)
{
    for i in 0...arxiusPdfs.count-1
    {
        if (editSel.contains(arxiusPdfs[i].nom))
        {

        }
        else
        {
            editSel.append(arxiusPdfs[i].nom)
        }
    }
    btnTrash.isHidden=false
    reload()
}

@IBAction func editClear(_ sender: AnyObject)
{
    for i in 0...arxiusPdfs.count-1
    {
        if (editSel.contains(arxiusPdfs[i].nom))
        {
            editSel.remove(at: editSel.index(of: arxiusPdfs[i].nom)!)
        }
    }
    btnTrash.isHidden=true
    reload()
}


//////// TAULA GENERAL ///////////////
```

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -
    > Int
{
    return arxiusPdfs.count
}

internal func tableView(_ tableView: UITableView, cellForRowAt indexPath:
    IndexPath) -> UITableViewCell
{
    let cell = UITableViewCell(style: UITableViewCellStyle.default,
        reuseIdentifier: "Cell")

    if (taulageneral.isEditing)
    {
        cell.textLabel?.text = arxiusPdfs[indexPath.row].nom
        cell.imageView!.image = UIImage(named: "icona")

        if (editSel.contains(arxiusPdfs[indexPath.row].nom))
        {
            tableView.selectRow(at: indexPath, animated: true, scrollPosition:
                UITableViewScrollPosition.none)
        }
    }
    else
    {

        cell.textLabel?.text = arxiusPdfs[indexPath.row].nom
        cell.imageView!.image = UIImage(named: "icona")

    }
    return cell
}

func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath)
{
    pdfSel=indexPath.row

    if (taulageneral.isEditing)
    {
        editSel.append(arxiusPdfs[indexPath.row].nom)

        if (editSel.count>0)
        {
            btnTrash.isHidden=false
        }

    }
    else
    {
        tableView.deselectRow(at: indexPath, animated: true)
        switch (seccioActiva)
        {
        case 1:
            self.performSegue(withIdentifier: "pre", sender: pdfSel)
            break

        case 2:
            self.performSegue(withIdentifier: "rec", sender: pdfSel)
            break

        case 3:
```

```
                self.performSegue(withIdentifier: "post", sender: pdfSel)
                break

            default:
                seccioActiva=1
                self.performSegue(withIdentifier: "pre", sender: pdfSel)
                break

            }

        }
    }


    func tableView(_ tableView: UITableView, didDeselectRowAt indexPath:
        IndexPath)
    {
        editSel.remove(at: editSel.index(of: arxiusPdfs[indexPath.row].nom)!)
        tableView.deselectRow(at: indexPath, animated: true)
        if (editSel.count==0)
        {
            btnTrash.isHidden=true
        }

    }


    func tableView(_ tableView: UITableView, commit editingStyle:
        UITableViewCellEditingStyle, forRowAt indexPath: IndexPath)
    {
        if editingStyle == UITableViewCellEditingStyle.delete
        {
            deletePDFfromCoreData(arxiusPdfs[indexPath.row].nom)
            arxiusPdfs.remove(at: indexPath.row)
            taulageneral.reloadData()
        }
    }


    func editTable (_ gestureRecognizer:UIGestureRecognizer)
    {
        if (gestureRecognizer.state == UIGestureRecognizerState.began)
        {
            taulageneral.isEditing=true
            btnDone.isHidden=false
            btnLogOut.isHidden=true
            btnLogIn.isHidden=true
            btnEditSelectAll.isHidden=false
            btnEditClear.isHidden=false
        }
    }


///// RELOAD TABLE ////


    func reloadTableData(_ notification: Notification)
    {
        reload()
    }

    func reload()
```

```
    {
        arxiusPdfs.removeAll()

        let appDel: AppDelegate = UIApplication.shared.delegate as! AppDelegate
        let context: NSManagedObjectContext = appDel.managedObjectContext
        let request = NSFetchRequest<NSFetchRequestResult>(entityName:
            "Documents")

        request.returnsObjectsAsFaults = false

        do
        {
            let results = try context.fetch(request)

            if (results.count>0)
            {
                for result in results as! [NSManagedObject]
                {
                    let arxiuProv = pdfs()
                    if let nom = result.value(forKey: "nom") as? String
                    {
                        arxiuProv.nom = nom
                    }
                    if let page = result.value(forKey: "currentPage") as? Int16
                    {
                        arxiuProv.page = page
                    }
                    if let take = result.value(forKey: "currentTake") as? Int16
                    {
                        arxiuProv.take = take
                    }
                    if let maxtake = result.value(forKey: "maxTake") as? Int16
                    {
                        arxiuProv.maxTake = maxtake
                    }
                    arxiusPdfs.append(arxiuProv)
                }
            }
        }
        catch
        {
            print("No s'ha carregat bé l'array")
        }


        taulageneral.reloadData()
    }

//// CANVI COLOR SECCIÓ

    func HomeSeccActiva ()
    {
        switch (seccioActiva)
        {
        case 0:
            btnPre.setTitleColor(UIColor.blue, for: UIControlState())
            btnRec.setTitleColor(UIColor.blue, for: UIControlState())
            btnPost.setTitleColor(UIColor.blue, for: UIControlState())

            btnPre.backgroundColor=UIColor.white
            btnRec.backgroundColor=UIColor.white
            btnPost.backgroundColor=UIColor.white
```

```
        break

        case 1:
            btnPre.setTitleColor(UIColor.white, for: UIControlState())
            btnRec.setTitleColor(UIColor.blue, for: UIControlState())
            btnPost.setTitleColor(UIColor.blue, for: UIControlState())

            btnPre.backgroundColor=UIColor.blue
            btnRec.backgroundColor=UIColor.white
            btnPost.backgroundColor=UIColor.white
        break

        case 2:
            btnPre.setTitleColor(UIColor.blue, for: UIControlState())
            btnRec.setTitleColor(UIColor.white, for: UIControlState())
            btnPost.setTitleColor(UIColor.blue, for: UIControlState())

            btnPre.backgroundColor=UIColor.white
            btnRec.backgroundColor=UIColor.blue
            btnPost.backgroundColor=UIColor.white
        break

        case 3:
            btnPre.setTitleColor(UIColor.blue, for: UIControlState())
            btnRec.setTitleColor(UIColor.blue, for: UIControlState())
            btnPost.setTitleColor(UIColor.white, for: UIControlState())

            btnPre.backgroundColor=UIColor.white
            btnRec.backgroundColor=UIColor.white
            btnPost.backgroundColor=UIColor.blue
        break

        default:
            btnPre.setTitleColor(UIColor.blue, for: UIControlState())
            btnRec.setTitleColor(UIColor.blue, for: UIControlState())
            btnPost.setTitleColor(UIColor.blue, for: UIControlState())

            btnPre.backgroundColor=UIColor.white
            btnRec.backgroundColor=UIColor.white
            btnPost.backgroundColor=UIColor.white
        break


        }
    }

////// DELETE CORE DATA PDF /////


    func deletePDFfromCoreData (_ nom:String)
    {
        let appDel: AppDelegate = UIApplication.shared.delegate as! AppDelegate
        let context: NSManagedObjectContext = appDel.managedObjectContext
        var request = NSFetchRequest<NSFetchRequestResult>(entityName:
            "Documents")

        request.predicate = NSPredicate(format: "nom = %@", nom)
        request.returnsObjectsAsFaults = false

        do
        {
            let results = try context.fetch(request)
```

```
        if (results.count>0)
        {
            for result in results as! [NSManagedObject]
            {

                context.delete(result)

                do
                {
                    try context.save()
                }
                catch
                {

                }

            }
        }
    }
    catch
    {
        print("No s'ha eliminat bé")
    }
    request = NSFetchRequest<NSFetchRequestResult>(entityName: "Anotacions")

    request.predicate = NSPredicate(format: "document = %@", nom)
    request.returnsObjectsAsFaults = false

    do
    {
        let results = try context.fetch(request)

        if (results.count>0)
        {
            for result in results as! [NSManagedObject]
            {

                context.delete(result)

                do
                {
                    try context.save()
                }
                catch
                {

                }

            }
        }
    }
    catch
    {
        print("No s'ha eliminat bé")
    }

}
```

A.4.2 Secció PRE

```
//
//  VisorViewController.swift
//  TFG-importFile
//
//  Created by Adrià Martínez on 2/11/15.
//  Copyright © 2015 Adrià Martínez. All rights reserved.
//

import UIKit

class PRE_ViewController: UIViewController
{

    @IBOutlet var btnSistemes: UIButton!


    override func viewDidLoad()
    {
        super.viewDidLoad()
        EnDisableSistemes(false)
    }


    @IBAction func pre(_ sender: AnyObject)
    {
        seccioActiva=1
    }


    @IBAction func rec(_ sender: AnyObject)
    {
        seccioActiva=2
    }


    @IBAction func post(_ sender: AnyObject)
    {
        seccioActiva=3
    }


    @IBAction func sistemes(_ sender: Any)
    {
        if sistemesON == true
        {
            sistemesON=false
            btnSistemes.alpha=1
            EnDisableSistemes(false)
        }
        else
        {
            sistemesON=true
            btnSistemes.alpha=0.5
            EnDisableSistemes(true)
        }

    }



    func EnDisableSistemes(_ trufes:Bool) //Activa o desactiva els sistemes
```

```
{
    for n in 0..<docNotes.count
    {
        if (docNotes[n].pag == arxiusPdfs[pdfSel].page)
        {
            if (docNotes[n].visible)
            {
                if (docNotes[n].nota==0)
                {
                    docNotes[n].boto.isEnabled = trufes
                }
            }

        }

    }
}


}
```

A.4.3 Secció REC

```
//
//  REC-ViewController.swift
//  TFG-importFile
//
//  Created by Adrià Martínez on 3/11/15.
//  Copyright © 2015 Adrià Martínez. All rights reserved.
//

import UIKit
import CoreData


class REC_ViewController: UIViewController {

    //Declaració dels botons i labels provinents de la UI gràfica.

    @IBOutlet var lblTake: UILabel!
    @IBOutlet var btnTake: UIButton!
    @IBOutlet var btnTakeIn: UIButton!
    @IBOutlet var btnTakeOut: UIButton!
    @IBOutlet var btnNota1: UIButton!
    @IBOutlet var btnNota2: UIButton!
    @IBOutlet var btnNota3: UIButton!
    @IBOutlet var btnNota4: UIButton!
    @IBOutlet var btnNota5: UIButton!

    @IBOutlet var imgBafarada: UIImageView!
    @IBOutlet var lblInsertNewTake: UILabel!
    @IBOutlet var lblFonsBafarada: UILabel!
    @IBOutlet var lblTakeNum: UITextField!
    @IBOutlet var btnDoneCanviTake: UIButton!

    var n = 0

    override func viewDidLoad()
    {
        sistemesON=false
        updateTake()

        btnNota1.isHidden=true
        btnNota2.isHidden=true
        btnNota3.isHidden=true
        btnNota4.isHidden=true
        btnNota5.isHidden=true

        imgBafarada.isHidden=true
        lblInsertNewTake.isHidden=true
        lblFonsBafarada.isHidden=true
        lblTakeNum.isHidden=true
        btnDoneCanviTake.isHidden=true

        btnNota1.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)
        btnNota1.titleLabel?.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)
        btnNota1.imageView?.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)

        btnNota2.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)
        btnNota2.titleLabel?.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)
        btnNota2.imageView?.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)

        btnNota3.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)
        btnNota3.titleLabel?.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)
```

```
        btnNota3.imageView?.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)

        btnNota4.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)
        btnNota4.titleLabel?.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)
        btnNota4.imageView?.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)

        btnNota5.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)
        btnNota5.titleLabel?.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)
        btnNota5.imageView?.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)

        let longGesture = UILongPressGestureRecognizer(target: self, action:
            #selector(REC_ViewController.canviTake))
        longGesture.minimumPressDuration = 1
        btnTake.addGestureRecognizer(longGesture)


        NotificationCenter.default.addObserver(self, selector: #selector
            (REC_ViewController.ChangeButtons(_:)), name: NSNotification.Name
            (rawValue: "ChangeButtonsON"), object: nil)

        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }

    @IBAction func pre(_ sender: AnyObject)
    {
        seccioActiva=1
    }


    @IBAction func rec(_ sender: AnyObject)
    {
        seccioActiva=2
    }


    @IBAction func post(_ sender: AnyObject)
    {
        seccioActiva=3
    }

    @IBAction func TakeMenys(_ sender: Any)
    {
        if arxiusPdfs[pdfSel].take > 1
        {
            arxiusPdfs[pdfSel].take = arxiusPdfs[pdfSel].take - 1
            btnTakeOut.isEnabled=false
            btnTakeIn.isEnabled=false
            takeInOn=false
            takeOutOn=false
            updateTake()
        }
    }

    @IBAction func TakeMes(_ sender: Any)
    {
        if (btnTakeOut.isEnabled)
        {

            let alert = UIAlertController(title: "Missing Take Out", message:
                "You must set the end of the take before creating new one. Do
```

```
                you want to set it?", preferredStyle:
                UIAlertControllerStyle.alert)
            alert.addAction(UIAlertAction(title: "Yes", style: .default, handler:
                { (action) -> Void in

                takeOutOn = true
                self.btnTakeOut.isEnabled=false

            }))

            alert.addAction(UIAlertAction(title: "No", style: .default, handler:
                { (action) -> Void in

            }))

            self.present(alert, animated: true, completion: nil)
        }
        else
        {
            arxiusPdfs[pdfSel].take = arxiusPdfs[pdfSel].take + 1
            updateTake()
        }

    }


    @IBAction func takeIn(_ sender: Any)
    {

    }

    @IBAction func takeOut(_ sender: Any)
    {
        takeOutOn = true
        btnTakeOut.isEnabled=false
    }

    @IBAction func ChangeNote(_ sender: AnyObject)
    {
        let buttonClicked = sender.tag!
        switch buttonClicked
        {
            case 1: docNotes[n].nota=1
            break
            case 2: docNotes[n].nota=2
            break
            case 3: docNotes[n].nota=3
            break
            case 4: docNotes[n].nota=4
            break
            case 5: docNotes[n].nota=5
            break
            default:
                print("default")
            break
        }
        updateNote(n)
        NotificationCenter.default.post(name: Notification.Name(rawValue:
            "refreshNotes"), object: nil) //Envia un avís a una funció de la
            classe "Visor"
    }
```

```swift
func ChangeButtons(_ notification: Notification)
{
    let userInfo = notification.userInfo //Rep la "n" des de la classe
        "Visor"
    n = userInfo?["n"] as! Int

    btnNota1.isHidden=false
    btnNota2.isHidden=false
    btnNota3.isHidden=false
    btnNota4.isHidden=false
    btnNota5.isHidden=false

    print (n)

}


func updateTake ()
{

    let appDel: AppDelegate = UIApplication.shared.delegate as! AppDelegate
    let context: NSManagedObjectContext = appDel.managedObjectContext
    let request = NSFetchRequest<NSFetchRequestResult>(entityName:
        "Documents")
    request.predicate = NSPredicate(format: "nom = %@",
        arxiusPdfs[pdfSel].nom) //Posa com a condició que el document sigui
        igual a l'arxiu que he triat

    do
    {
        let results = try context.fetch(request)

        if (results.count>0)
        {
            let managedObject = results[0]
            (managedObject as AnyObject).setValue(arxiusPdfs[pdfSel].take,
                forKey: "currentTake")
            (managedObject as AnyObject).setValue(arxiusPdfs[pdfSel].maxTake,
                forKey: "maxTake")

            do
            {
                try context.save()
            }
            catch
            {

            }

        }
    }
    catch
    {
        print("fail")
    }

    lblTake.text = String(arxiusPdfs[pdfSel].take)
    btnTakeIn.setTitle(String(arxiusPdfs[pdfSel].take),for: .normal)
    btnTakeOut.setTitle(String(arxiusPdfs[pdfSel].take),for: .normal)

    btnNota1.setTitle(String(arxiusPdfs[pdfSel].take),for: .normal)
```

```
    btnNota2.setTitle(String(arxiusPdfs[pdfSel].take),for: .normal)
    btnNota3.setTitle(String(arxiusPdfs[pdfSel].take),for: .normal)
    btnNota4.setTitle(String(arxiusPdfs[pdfSel].take),for: .normal)
    btnNota5.setTitle(String(arxiusPdfs[pdfSel].take),for: .normal)

    takeInOn=true
    takeOutOn=false
    btnTakeIn.isEnabled=false
    btnTakeOut.isEnabled=true
    if (arxiusPdfs[pdfSel].maxTake<arxiusPdfs[pdfSel].take)
    {
        arxiusPdfs[pdfSel].maxTake = arxiusPdfs[pdfSel].take
    }
    else
    {
        for n in 0..<docNotes.count
        {
            if docNotes[n].take==arxiusPdfs[pdfSel].take
            {
                if docNotes[n].nota==98
                {
                    takeInOn=false
                }
                if docNotes[n].nota==99
                {
                    btnTakeOut.isEnabled=false
                }

            }
        }
    }
    disableButtons()
}

func canviTake()
{

    imgBafarada.isHidden=false
    lblInsertNewTake.isHidden=false
    lblFonsBafarada.isHidden=false
    lblTakeNum.isHidden=false
    btnDoneCanviTake.isHidden=false
}

@IBAction func btnCanviTake(_ sender: Any)
{
    if ((lblTakeNum) != nil)
    {
        arxiusPdfs[pdfSel].take = Int16(lblTakeNum.text!)!
        lblTake.text=lblTakeNum.text!
    }
    imgBafarada.isHidden=true
    lblInsertNewTake.isHidden=true
    lblFonsBafarada.isHidden=true
    lblTakeNum.isHidden=true
    btnDoneCanviTake.isHidden=true

    updateTake()

    view.endEditing(true)
}
```

```
    func disableButtons()
    {
        for n in 0..<docNotes.count
        {
            print (docNotes.count)
            if (docNotes[n].pag == arxiusPdfs[pdfSel].page)
            {
                if (docNotes[n].visible)
                {
                    if (docNotes[n].take==arxiusPdfs[pdfSel].take)
                    {
                        docNotes[n].boto.isEnabled = true
                        docNotes[n].boto.setTitleColor(UIColor.blue, for: .normal)
                    }
                    else
                    {
                        docNotes[n].boto.isEnabled = false
                    }
                }

            }

        }


    }

}
```

A.4.4 Secció POST

```
//
//  POST-ViewController.swift
//  TFG-importFile
//
//  Created by Adrià Martínez on 3/11/15.
//  Copyright © 2015 Adrià Martínez. All rights reserved.
//

import UIKit

class POST_ViewController: UIViewController
{

    override func viewDidLoad()
    {
        sistemesON=false
        for n in 0..<docNotes.count
        {
            if (docNotes[n].pag == arxiusPdfs[pdfSel].page) //Desactiva les tots
                els botons
            {
                docNotes[n].boto.isEnabled = false
            }
        }
        super.viewDidLoad()
    }


    @IBAction func pre(_ sender: AnyObject)
    {
        seccioActiva=1
    }

    @IBAction func rec(_ sender: AnyObject)
    {
        seccioActiva=2
    }

    @IBAction func post(_ sender: AnyObject)
    {
        seccioActiva=3
    }


}
```

A.4.5 Visor

```
//
//  PRE-Visor.swift
//  TFG-importFile
//
//  Created by Adrià Martínez on 2/11/15.
//  Copyright © 2015 Adrià Martínez. All rights reserved.
//

import UIKit
import CoreData
import CoreGraphics
import M13PDFKit


class Visor: PDFKBasicPDFViewer
{

    override func viewDidLoad()
    {
        super.viewDidLoad()

        if (arxiusPdfs.isEmpty)
        {
            print("No hi ha arxius per mostrar")
        }
        else
        {

            var docURL = (FileManager.default.urls(for: .documentDirectory, in: .
                userDomainMask)).last as? NSURL
            docURL = docURL?.appendingPathComponent(arxiusPdfs[pdfSel].nom) as
                NSURL?


            //Crea el document i el mostra per pantalla

            let document = PDFKDocument(contentsOfFile:docURL?.path, password: nil)
            self.loadDocument(document)



            downloadNotes() //PER FER: Mirar abans si hem canviat de document
        }

        NotificationCenter.default.addObserver(self, selector: #selector(Visor.
            refreshNotes(_:)), name: NSNotification.Name(rawValue:
            "refreshNotes"), object: nil)


// Declaraions dels gestos

        let tap1 = UITapGestureRecognizer(target: self, action: #selector(self.
            TapFunc1(_:)))
        tap1.numberOfTapsRequired = 1
        self.view.addGestureRecognizer(tap1)

        let tap2 = UITapGestureRecognizer(target: self, action: #selector(self.
            TapFunc2(_:)))
        tap2.numberOfTapsRequired = 2
        self.view.addGestureRecognizer(tap2)
```

```
        let tap3 = UITapGestureRecognizer(target: self, action: #selector(self.
            TapFunc3(_:)))
        tap3.numberOfTapsRequired = 3
        self.view.addGestureRecognizer(tap3)

        let tap4 = UITapGestureRecognizer(target: self, action: #selector(self.
            TapFunc4(_:)))
        tap4.numberOfTapsRequired = 4
        self.view.addGestureRecognizer(tap4)

        let tap5 = UITapGestureRecognizer(target: self, action: #selector(self.
            TapFunc5(_:)))
        tap5.numberOfTapsRequired = 5
        self.view.addGestureRecognizer(tap5)


        let longPressPost = UILongPressGestureRecognizer(target: self, action:
            #selector(self.QuinesPresesAqui(_:)))
        longPressPost.minimumPressDuration = 1
        self.view.addGestureRecognizer(longPressPost)

    }

    override func viewDidAppear(_ animated: Bool)
    {
        drawPageNotes() //Quan apareix la "vista" es pinten les notes
            d'aquella pàgina
    }


    var isMoving = false

    ///////////////////////////////////////////////////////////////
    /////////////////////  TAP'S //////////////////////////////
    ///////////////////////////////////////////////////////////////


//////////// TAP 1 //////////////

    func TapFunc1(_ sender: UIGestureRecognizer)
    {
        if sender is UITapGestureRecognizer
        {
            if sistemesON==false
            {
                if (arxiusPdfs[pdfSel].page != Int16(self.document.currentPage))
                {
                    arxiusPdfs[pdfSel].page = Int16(self.document.currentPage)
                    for n in 0..<docNotes.count
                    {
                        docNotes[n].boto.removeFromSuperview()
                    }
                    drawPageNotes()

                }
                else
                {
                    switch (seccioActiva)
                    {
                    case 1:
                        print ("ei1 del pre")
```

```
                        break
                    case 2:
                        var note = Int16()
                        if (takeInOn)
                        {
                            note = 98
                        }
                        else if (takeOutOn)
                        {
                            note = 99
                        }
                        else
                        {
                            note = 1
                        }
                        let punt = sender.location(in: self.view)
                        let sistema = quinSistema(Float(punt.y))
                        print(sistema)
                        saveNote(nota: note, pagina:
                            Int16(self.document.currentPage), x: Float(punt.x), y:
                            Float(punt.y), sistema:sistema, take:
                            arxiusPdfs[pdfSel].take)
                        drawNote(Float(punt.x), y: Float(punt.y), note: note, n:
                            docNotes.count-1, take: arxiusPdfs[pdfSel].take)
                        NotificationCenter.default.post(name: Notification.Name
                            (rawValue: "ChangeButtonsON"), object: nil, userInfo:
                            ["n":docNotes.count-1])
                        break

                    case 3:
                        print ("ei1 del post")
                        break
                    default:
                        print ("ei1 default")
                        break
                    }

                }
            }

        }

    }


//////////// TAP 2 //////////////

    func TapFunc2(_ sender: UIGestureRecognizer)
    {
        if let tapped = sender as? UITapGestureRecognizer
        {
            if sistemesON==true
            {
                let punt = tapped.location(in: self.view)
                saveNote(nota: 0, pagina: Int16(self.document.currentPage), x: 0,
                    y: Float(punt.y), sistema: 0, take: 0)
                reloadSistemes()
            }
            else
            {
                switch (seccioActiva)
```

```
                {
                case 1:
                    print ("ei2 del pre")
                    break
                case 2:
                    print ("ei2 del rec")
                    let punt = sender.location(in: self.view)
                    let sistema = quinSistema(Float(punt.y))
                    deleteNote(docNotes.count-1)
                    saveNote(nota: 2, pagina: Int16(self.document.currentPage), x:
                        Float(punt.x), y: Float(punt.y), sistema:sistema, take:
                        arxiusPdfs[pdfSel].take)
                    drawNote(Float(punt.x), y: Float(punt.y), note: 2, n: docNotes.
                        count-1, take: arxiusPdfs[pdfSel].take)
                    NotificationCenter.default.post(name: Notification.Name
                        (rawValue: "ChangeButtonsON"), object: nil, userInfo:
                        ["n":docNotes.count-1])
                    break
                case 3:
                    print ("ei2 del post")
                    break
                default:
                    print ("ei2 default")
                    break
                }
            }
        }
    }


/////////////  TAP 3 ///////////////

    func TapFunc3(_ sender: UIGestureRecognizer)
    {
        if sender is UITapGestureRecognizer
        {
            if sistemesON==false
            {

                switch (seccioActiva)
                {
                case 1:
                    print ("ei3 del pre")
                    break
                case 2:
                    print ("ei3 del rec")
                    let punt = sender.location(in: self.view)
                    let sistema = quinSistema(Float(punt.y))
                    print(sistema)
                    deleteNote(docNotes.count-1)
                    saveNote(nota: 3, pagina: Int16(self.document.currentPage), x:
                        Float(punt.x), y: Float(punt.y), sistema:sistema, take:
                        arxiusPdfs[pdfSel].take)
                    drawNote(Float(punt.x), y: Float(punt.y), note: 3, n: docNotes.
                        count-1, take: arxiusPdfs[pdfSel].take)
                    NotificationCenter.default.post(name: Notification.Name
                        (rawValue: "ChangeButtonsON"), object: nil, userInfo:
                        ["n":docNotes.count-1])
                    break
                case 3:
```

```
                print ("ei3 del post")
                break
            default:
                print ("ei3 default")
                break
            }

        }
    }

}


///////////// TAP 4 //////////////

func TapFunc4(_ sender: UIGestureRecognizer)
{
    if sender is UITapGestureRecognizer
    {
        if sistemesON==false
        {

            switch (seccioActiva)
            {
            case 1:
                print ("ei4 del pre")
                break
            case 2:
                print ("ei4 del rec")
                let punt = sender.location(in: self.view)
                let sistema = quinSistema(Float(punt.y))
                print(sistema)
                deleteNote(docNotes.count-1)
                saveNote(nota: 4, pagina: Int16(self.document.currentPage), x:
                    Float(punt.x), y: Float(punt.y), sistema:sistema, take:
                    arxiusPdfs[pdfSel].take)
                drawNote(Float(punt.x), y: Float(punt.y), note: 4, n: docNotes.
                    count-1, take: arxiusPdfs[pdfSel].take)
                NotificationCenter.default.post(name: Notification.Name
                    (rawValue: "ChangeButtonsON"), object: nil, userInfo:
                    ["n":docNotes.count-1])
                break
            case 3:
                print ("ei4 del post")
                break
            default:
                print ("ei4 default")
                break
            }

        }

    }

}


///////////// TAP 5 //////////////

func TapFunc5(_ sender: UIGestureRecognizer)
```

```
    {
        if sender is UITapGestureRecognizer
        {
            if sistemesON==false
            {

                switch (seccioActiva)
                {
                case 1:
                    print ("ei5 del pre")
                    break
                case 2:
                    print ("ei5 del rec")
                    let punt = sender.location(in: self.view)
                    let sistema = quinSistema(Float(punt.y))
                    print(sistema)
                    deleteNote(docNotes.count-1)
                    saveNote(nota: 5, pagina: Int16(self.document.currentPage), x:
                        Float(punt.x), y: Float(punt.y), sistema:sistema, take:
                        arxiusPdfs[pdfSel].take)
                    drawNote(Float(punt.x), y: Float(punt.y), note: 5, n: docNotes.
                        count-1, take: arxiusPdfs[pdfSel].take)
                    NotificationCenter.default.post(name: Notification.Name
                        (rawValue: "ChangeButtonsON"), object: nil, userInfo:
                        ["n":docNotes.count-1])
                    break
                case 3:
                    print ("ei5 del post")
                    break
                default:
                    print ("ei5 default")
                    break
                }

            }

        }

    }





    ///////////////////////////////////////////////////////////
    ////////////////////  BOTONS ////////////////////////////
    ///////////////////////////////////////////////////////////


/// Funció que es crida quan es toca un botó. El sender.tag determina quin
    és el botó que s'ha tocat (coincideix amb la posició de l'array)
    func btnsSistemes(_ sender: AnyObject)
    {
        let n = Int(sender.tag)
        print(n)
    }
```

```
/// Funció que es crida quan es mou un botó. El sender.tag determina quin és
    el botó que s'ha mogut (coincideix amb la posició de l'array)
    func wasDragged (sender : UIButton, event :UIEvent)
    {
        isMoving = true
        let n = sender.tag
        _ = sender as UIView;
        let touches : Set<UITouch> = event.touches(for: sender)!
        let touch = touches.first!
        let location = touch.location(in: self.view)
        if docNotes[n].nota==0
        {
            docNotes[n].boto.center = CGPoint(x: self.view.center.x, y: location.y)
            docNotes[n].y=Float(location.y)
        }
        else
        {
            docNotes[n].boto.center = CGPoint(x: location.x, y: location.y)
            docNotes[n].y=Float(location.y)
        }

    }



/// Funció que es crida quan es deixa anar un botó. El sender.tag determina
    quin és el botó que s'ha deixat anar (coincideix amb la posició de
    l'array)
    func wasCancelled (sender : UIButton, event :UIEvent)
    {
        isMoving=false
        let n = sender.tag
        updateNote(n)
        reloadSistemes()
    }



/// Funció que es crida quan es toca un botó durant un segon. El sender.tag
    determina quin és el botó que s'ha deixat anar (coincideix amb la posició
    de l'array)

    func longPress (_ sender: UIGestureRecognizer)
    {
        if (isMoving==false)
        {
            if sender.state == .began //Si no hi hagués això, es cridaria també
                al deixar anar el botó
            {
                if let button = sender.view as? UIButton //Creem un botó per
                    poder saber quin és el cridat
                {
                    let n = button.tag

                    if (docNotes[n].nota==0) //En el cas que sigui un sistema,
                        l'eliminem directament (amb previ avís)
```

```
                    {
                        let alert = UIAlertController(title: "Delete staff
                            separation", message: "Are you sure you want to
                            delete that staff?", preferredStyle:
                            UIAlertControllerStyle.alert)
                        alert.addAction(UIAlertAction(title: "Yes",
                            style: .default, handler: { (action) -> Void in

                            deleteNote(n)
                            self.reloadSistemes()

                        }))

                        alert.addAction(UIAlertAction(title: "No", style: .default,
                            handler: { (action) -> Void in

                        }))

                        self.present(alert, animated: true, completion: nil)

                    }
                }
            }

        }
    }


    func recLongPress (_ sender: UIGestureRecognizer)
    {
        if (isMoving==false)
        {
            if sender.state == .began //Si no hi hagués això, es cridaria també
                al deixar anar el botó
            {
                if let button = sender.view as? UIButton //Creem un botó per
                    poder saber quin és el cridat
                {
                    let n = button.tag

                    NotificationCenter.default.post(name: Notification.Name
                        (rawValue: "ChangeButtonsON"), object: nil, userInfo:
                        ["n":n])

                }
            }

        }

    }


    func QuinesPresesAqui (_ sender: UIGestureRecognizer)
    {
        if (seccioActiva == 3)
        {
            let punt = sender.location(in: self.view)
            let sistemaPunt = quinSistema(Float(punt.y))
            if sender.state == .began
            {
                print ("Obrir popUp")
```

```
            var possiblesTakes = [Int16()]
            possiblesTakes.removeAll()
            for n in 0..<docNotes.count
            {
                if (docNotes[n].nota==98) //Busco les anotacions d'inici de
                    presa
                {

                    if (docNotes[n].sistema < sistemaPunt || (docNotes[n].
                        sistema == sistemaPunt && docNotes[n].x <
                        Float(punt.x)))
                    {
                       possiblesTakes.append(docNotes[n].take) //Em quedo les
                           preses que comencen abans que el punt tocat
                    }

                }
                for i in 0..<possiblesTakes.count
                {
                    //print("Numero de i: " +
                        String(possiblesTakes.count))
                    if (docNotes[n].nota==99 &&
                        docNotes[n].take==possiblesTakes[i]) //Busco el final
                        de presa dels takes que m'he quedat
                    {
                        if (docNotes[n].sistema > sistemaPunt || (docNotes[n].
                            sistema == sistemaPunt && docNotes[n].x >
                            Float(punt.x)))
                        {
                            print(docNotes[n].take) //Em quedo amb els que
                                acaben més tard que el punt
                        }
                        else
                        {
                            possiblesTakes.remove(at: i) // I elimino els que
                                acaben abans
                        }
                    }
                }

                }
        }
        if sender.state == .ended
        {
            print("Tancar popUp")
        }
    }

}



///////////////////////////////////////////////////////////
////////////////////  FUNCIONS /////////////////////////
///////////////////////////////////////////////////////////




// Funció que transforma la coordenada "y" d'un punt al número de sistema
    de la partitura (prèviament senyalitzat)
```

```
func quinSistema(_ punt:Float) ->Int16
{
    var sistema = 0
    for n in 0 ..< sMax.count
    {
        let range:Range = Float(sMin[n])..<Float(sMax[n])
        let iHaveIt = range.contains(Float(punt))
        if (iHaveIt == true)
        {
            sistema = n+1
        }
    }
    return Int16(sistema)
}




///// Funció per reordenar els sistemes abans de tornar-los a pintar
    (quan s'elimina o es crea un de nou)
func reloadSistemes()
{
    sMax.removeAll()
    sMin.removeAll()

    for n in 0..<docNotes.count
    {
        if (docNotes[n].pag == arxiusPdfs[pdfSel].page)
        {
            if (docNotes[n].nota==0)
            {
                docNotes[n].boto.removeFromSuperview()
            }
        }

    }

    var newYs = [NewYs()]
    newYs.removeAll()

    for n in 0..<docNotes.count
    {
        if (docNotes[n].pag == arxiusPdfs[pdfSel].page)
        {
            if (docNotes[n].visible)
            {
                if (docNotes[n].nota==0)
                {
                    let newY = NewYs()
                    newY.n=n
                    newY.y=docNotes[n].y
                    newYs.append(newY)
                }
            }

        }

    }
    let sortedNewYs = newYs.sorted{$0.y < $1.y} // Ordeno els arrays abans
        d'assignar els mínims i màxims ja que per contra, si afegís un
```

```
        sistema a la meitat de la pàgina estaria establint malament els
        seus mínims i màxims

    for n in 0..<newYs.count
    {
        nou6tema(sortedNewYs[n].y, n: sortedNewYs[n].n)
    }


}


//// Funció per crear un nou sistema (amb els corresponents botons)
func nou6tema (_ y:Float, n:Int)
{
    docNotes[n].boto.frame = CGRect(x: 0, y: Int(y), width: 972, height: 40)
    docNotes[n].boto.center = CGPoint(x: self.view.center.x, y: CGFloat(y))
    docNotes[n].boto.setImage(UIImage(named: "separacio-sistemes.png"), for: .
        normal)
    docNotes[n].boto.setImage(UIImage(named: "separacioOff.png"), for: .
        disabled)
    docNotes[n].boto.tag=n
    docNotes[n].boto.addTarget(self, action: #selector(Visor.btnsSistemes(_:)),
        for: UIControlEvents.touchDown)
    docNotes[n].boto.addTarget(self, action: #selector(Visor.wasDragged(sender:
        event:)), for: UIControlEvents.touchDragInside)
    docNotes[n].boto.addTarget(self, action: #selector(Visor.wasCancelled
        (sender:event:)), for: UIControlEvents.touchUpInside)
    docNotes[n].boto.addTarget(self, action: #selector(Visor.wasCancelled
        (sender:event:)), for: UIControlEvents.touchUpOutside)
    let longGesture = UILongPressGestureRecognizer(target: self, action:
        #selector(Visor.longPress(_:)))
    longGesture.minimumPressDuration = 1
    docNotes[n].boto.addGestureRecognizer(longGesture)
    self.view.addSubview(docNotes[n].boto)

    if (sMax.isEmpty)
    {
        sMin.append(0)
        sMax.append(y)
        sMin.append(y)
        sMax.append(9999)
    }
    else
    {
        sMax[sMax.count-1] = y
        sMin.append(y)
        sMax.append(9999)
    }

}


func drawNote (_ x:Float, y:Float, note:Int16, n:Int, take:Int16)
{

    docNotes[n].boto.tag=n
    docNotes[n].boto.setTitle(String(take),for: .normal)
    docNotes[n].boto.setTitleColor(UIColor.blue, for: .normal)
    docNotes[n].boto.setTitleColor(UIColor.gray, for: .disabled)
    docNotes[n].boto.setBackgroundImage(nil, for: .normal)
    docNotes[n].boto.setImage(nil, for: .normal)
```

```
docNotes[n].boto.transform = CGAffineTransform(scaleX: -1.0, y: 1.0)
docNotes[n].boto.titleLabel?.transform = CGAffineTransform(scaleX: -1.0, y:
    1.0)
docNotes[n].boto.imageView?.transform = CGAffineTransform(scaleX: -1.0, y:
    1.0)
docNotes[n].boto.frame = CGRect(x: Int(x), y: Int(y), width: 60, height: 20)

let longGesture = UILongPressGestureRecognizer(target: self, action:
    #selector(Visor.recLongPress(_:)))
longGesture.minimumPressDuration = 0.5
docNotes[n].boto.addGestureRecognizer(longGesture)
self.view.addSubview(docNotes[n].boto)

switch note
{

case 1:
    if (take<20)
    {
        docNotes[n].boto.frame = CGRect(x: Int(x), y: Int(y), width: 20,
            height: 20)
    }
    else if (take<150)
    {
        docNotes[n].boto.frame = CGRect(x: Int(x), y: Int(y), width: 30,
            height: 20)
    }
    else if (take>150)
    {
        docNotes[n].boto.frame = CGRect(x: Int(x), y: Int(y), width: 40,
            height: 20)
    }

    docNotes[n].boto.setBackgroundImage(UIImage(named: "Nota1.png"), for: .
        normal)
    docNotes[n].boto.titleLabel?.textAlignment = NSTextAlignment.center
    break

case 2:

    docNotes[n].boto.setImage(UIImage(named: "Nota2.png"), for: .normal)
    break
case 3:

    docNotes[n].boto.setImage(UIImage(named: "Nota3.png"), for: .normal)
    break

case 4:

    docNotes[n].boto.setImage(UIImage(named: "Nota4.png"), for: .normal)
    break

case 5:

    docNotes[n].boto.setImage(UIImage(named: "Nota5.png"), for: .normal)
    break

case 98:
    if (take<150)
    {
        docNotes[n].boto.frame = CGRect(x: Int(x), y: Int(y), width: 30,
```

```
                height: 30)
        }
        else
        {
            docNotes[n].boto.frame = CGRect(x: Int(x), y: Int(y), width: 60,
                height: 30)
        }
        docNotes[n].boto.transform = CGAffineTransform(scaleX: 1, y: 1.0)
        docNotes[n].boto.titleLabel?.transform = CGAffineTransform(scaleX: 1, y:
            1.0)
        docNotes[n].boto.imageView?.transform = CGAffineTransform(scaleX: 1, y:
            1.0)

        docNotes[n].boto.setBackgroundImage(UIImage(named: "TakeIn.png"),
            for: .normal)
        docNotes[n].boto.titleLabel?.textAlignment = NSTextAlignment.center
        takeInOn=false

        break


    case 99:

        if (take<150)
        {
            docNotes[n].boto.frame = CGRect(x: Int(x), y: Int(y), width: 30,
                height: 30)

        }
        else
        {
            docNotes[n].boto.frame = CGRect(x: Int(x), y: Int(y), width: 60,
                height: 30)
        }

        docNotes[n].boto.transform = CGAffineTransform(scaleX: 1, y: 1.0)
        docNotes[n].boto.titleLabel?.transform = CGAffineTransform(scaleX: 1, y:
            1.0)
        docNotes[n].boto.imageView?.transform = CGAffineTransform(scaleX: 1, y:
            1.0)

        docNotes[n].boto.setBackgroundImage(UIImage(named: "TakeOut.png"),
            for: .normal)
        docNotes[n].boto.titleLabel?.textAlignment = NSTextAlignment.center
        takeOutOn=false

        break


    default:
        break
    }

    docNotes[n].boto.center = CGPoint(x: CGFloat(x), y: CGFloat(y))
    self.view.addSubview(docNotes[n].boto)
}

//// Funció per pintar a la pantalla les notes d'aquella pàgina
func drawPageNotes()
{
```

```
    for n in 0..<docNotes.count
    {
        if (docNotes[n].pag == arxiusPdfs[pdfSel].page && docNotes[n].visible &&
            docNotes[n].nota != 0)
        {
            docNotes[n].boto.removeFromSuperview()
            drawNote(docNotes[n].x, y: docNotes[n].y, note: docNotes[n].nota, n:
                n, take:docNotes[n].take)
        }
    }
    reloadSistemes()
}


func refreshNotes(_ notification: Notification)
{
    drawPageNotes()
}






//////////////////////////////////////////////////////////
/////////////////  BASE DE DADES ////////////////////////
//////////////////////////////////////////////////////////


/// Funció per guardar una anotació a la base de dades i a l'array de la
    RAM. Necessita rebre el tipus de nota, les seves coordenades (x,y),
    la pagina, el sistema i la presa




//// Funció per descarregar les anotacions d'aquell document de la base
    de dades a la memòria RAM.
func downloadNotes ()
{

    docNotes.removeAll()

    let appDel: AppDelegate = UIApplication.shared.delegate as! AppDelegate
    let context: NSManagedObjectContext = appDel.managedObjectContext
    let request = NSFetchRequest<NSFetchRequestResult>(entityName:
        "Anotacions")

    request.predicate = NSPredicate(format: "document = %@",
        arxiusPdfs[pdfSel].nom)
    request.returnsObjectsAsFaults = false

    do
    {
        let results = try context.fetch(request)

        if (results.count>0)
        {
            for result in results as! [NSManagedObject]
            {
```

```
                        let currentNote = Notes()
                        if let i = result.value(forKey: "id") as? String
                        {
                            currentNote.id=i
                        }
                        if let i = result.value(forKey: "nota") as? Int16
                        {
                            currentNote.nota=i
                        }

                        if let i = result.value(forKey: "sistema") as? Int16
                        {
                            currentNote.sistema=i
                        }

                        if let i = result.value(forKey: "pagina") as? Int16
                        {
                            currentNote.pag=i
                        }

                        if let i = result.value(forKey: "take") as? Int16
                        {
                            currentNote.take=i
                        }

                        if let i = result.value(forKey: "x") as? Float
                        {
                            currentNote.x=i
                        }

                        if let i = result.value(forKey: "y") as? Float
                        {
                            currentNote.y=i
                        }

                        docNotes.append(currentNote)
                    }
                }
            }
            catch
            {
                print("fail")
            }

        }






    }
```

A.4.6 Funcions globals

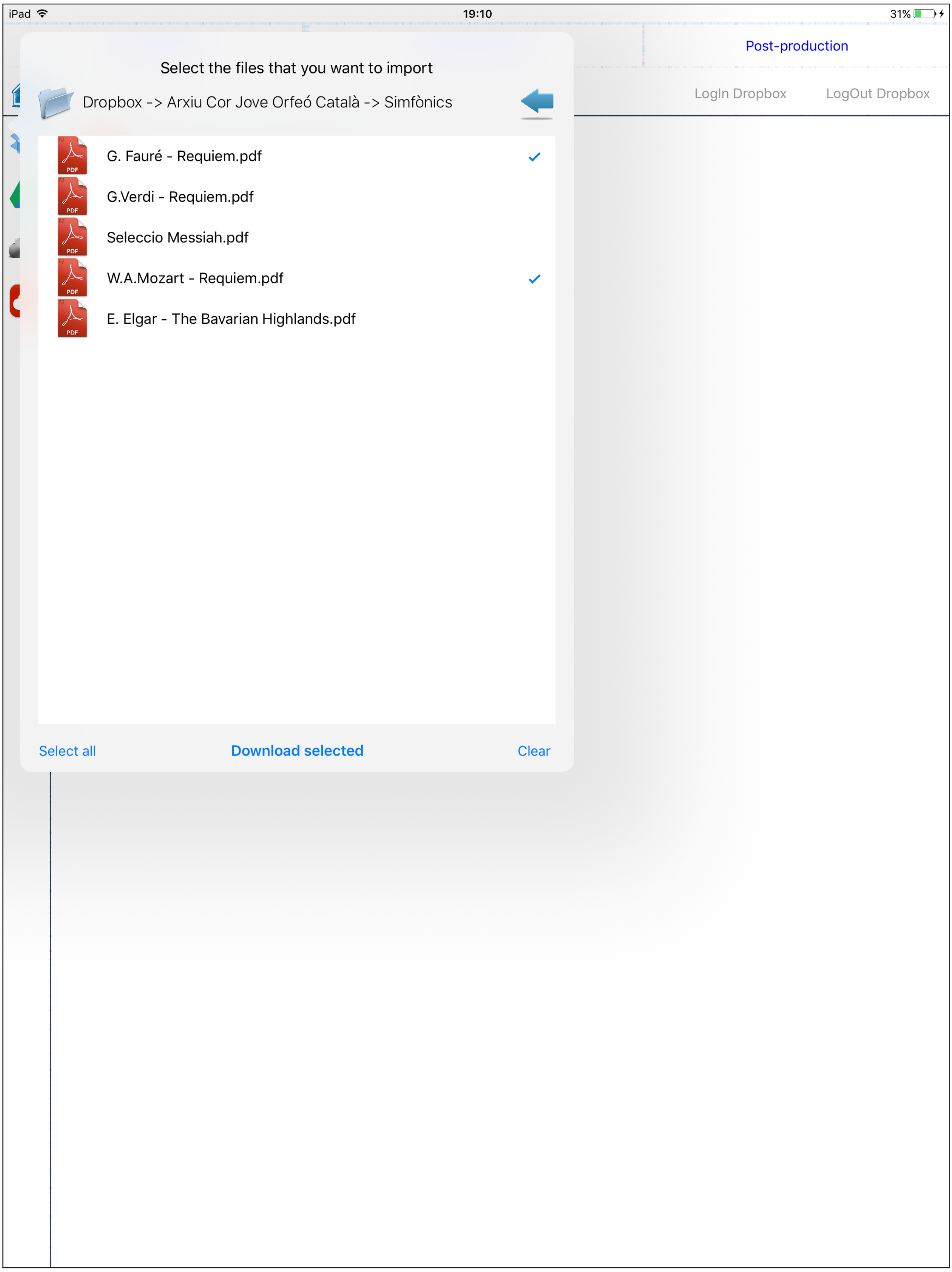
```
//
//  FuncionsGlobals.swift
//  MeisterNotes
//
//  Created by Adrià Martínez on 10/2/17.
//  Copyright © 2017 Adrià Martínez. All rights reserved.
//

import Foundation
import UIKit
import CoreData


////////////////////////////////////////////////////////////////
/////////////// VARIABLES GLOBALS //////////////////////
////////////////////////////////////////////////////////////////


var carpetesAMostrar = [""]
var escarpeta = [true]
var carpetesBaixades = false
var arxiusPdfs = [pdfs]()
var seccioActiva = 0
var pdfSel = 0;
var sistemesON = false
var takeInOn = false
var takeOutOn = false
var docNotes = [Notes()]
var sMin = [Float()]
var sMax = [Float()]

////////////////////////////////////////////////////////////////
/////////////////  CLASSES PROPIES //////////////////////
////////////////////////////////////////////////////////////////


class Notes
{
    var pag = Int16()
    var sistema = Int16()
    var take = Int16()
    var nota = Int16()
    var x = Float()
    var y = Float()
    var id = String()
    var boto = UIButton()
    var visible = true
}


class NewYs
{
    var y = Float()
    var n = Int()
}

class pdfs
{
    var nom = String()
```

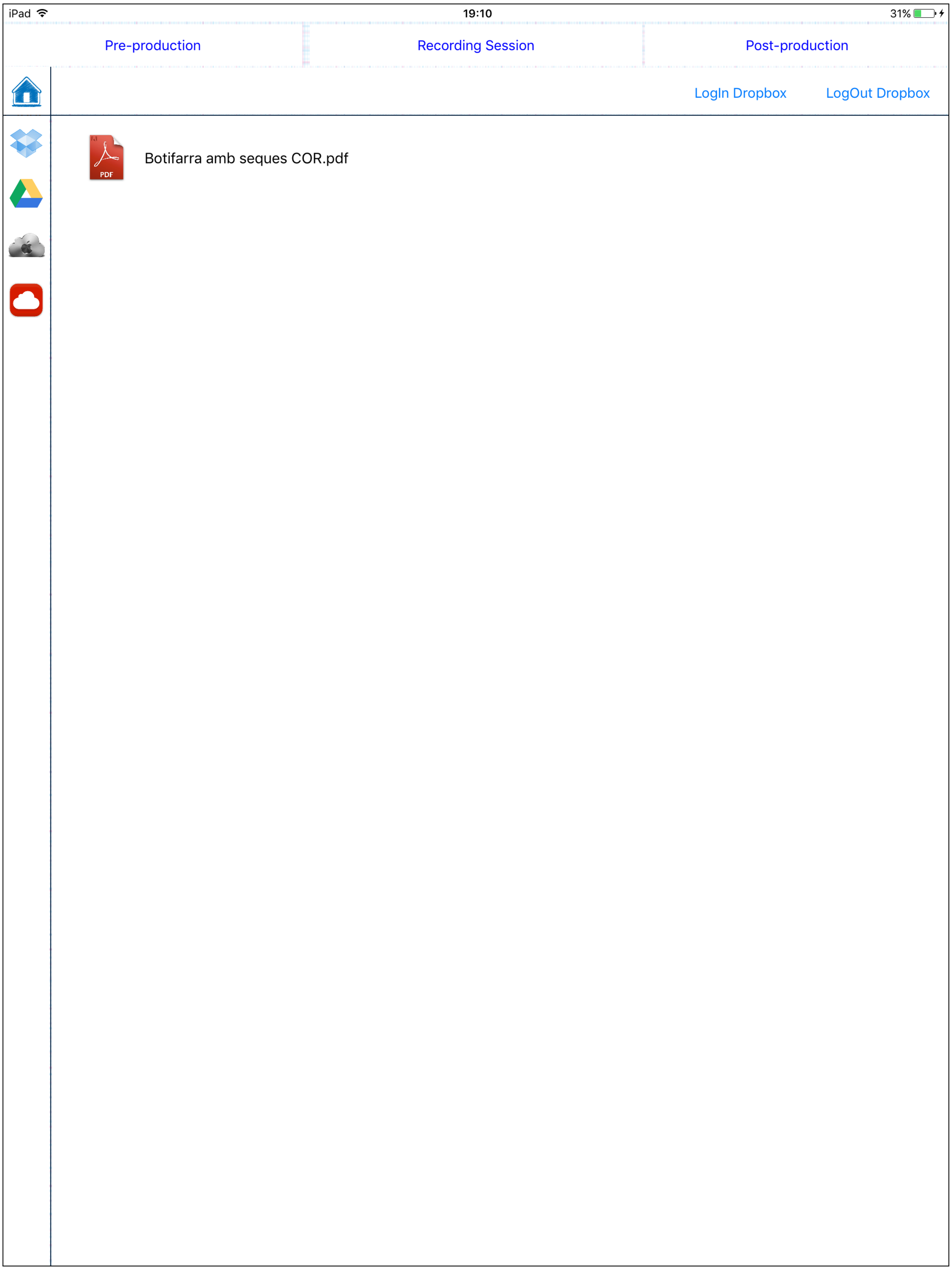
```
    var page = Int16()
    var take = Int16()
    var maxTake:Int16 = 0
}


////////////////////////////////////////////////////////////
//////////////// FUNCIONS GLOBALS ////////////////////////
////////////////////////////////////////////////////////////

func saveNote(nota:Int16, pagina:Int16, x:Float, y:Float, sistema:Int16, take:
    Int16)
{
    let appDel: AppDelegate = UIApplication.shared.delegate as! AppDelegate
    let context: NSManagedObjectContext = appDel.managedObjectContext


    let newEntry = NSEntityDescription.insertNewObject(forEntityName:
        "Anotacions", into: context)

    newEntry.setValue(arxiusPdfs[pdfSel].nom, forKey: "document")
    newEntry.setValue(nota, forKey: "nota")
    newEntry.setValue(pagina, forKey: "pagina")
    newEntry.setValue(sistema, forKey: "sistema")
    newEntry.setValue(take, forKey: "take")
    newEntry.setValue(x, forKey: "x")
    newEntry.setValue(y, forKey: "y")
    let id = String(describing: newEntry.objectID.hashValue)
    newEntry.setValue(id, forKey: "id")

    let currentNote = Notes()
    currentNote.id=id
    currentNote.nota=nota
    currentNote.pag=pagina
    currentNote.sistema=sistema
    currentNote.take=take
    currentNote.x=x
    currentNote.y=y

    docNotes.append(currentNote)

    do
    {
        try context.save()
        print ("-----------------------")
        print("ID: \(id)")
        print("Nota: \(nota)")
        print("Pàgina: \(pagina)")
        print("Sistema: \(sistema)")
        print("Take: \(take)")
        print("x: \(x)")
        print("y: \(y)")
        print ("-----------------------")
    }
    catch
    {
        print("No s'ha desat bé")
    }
}
```

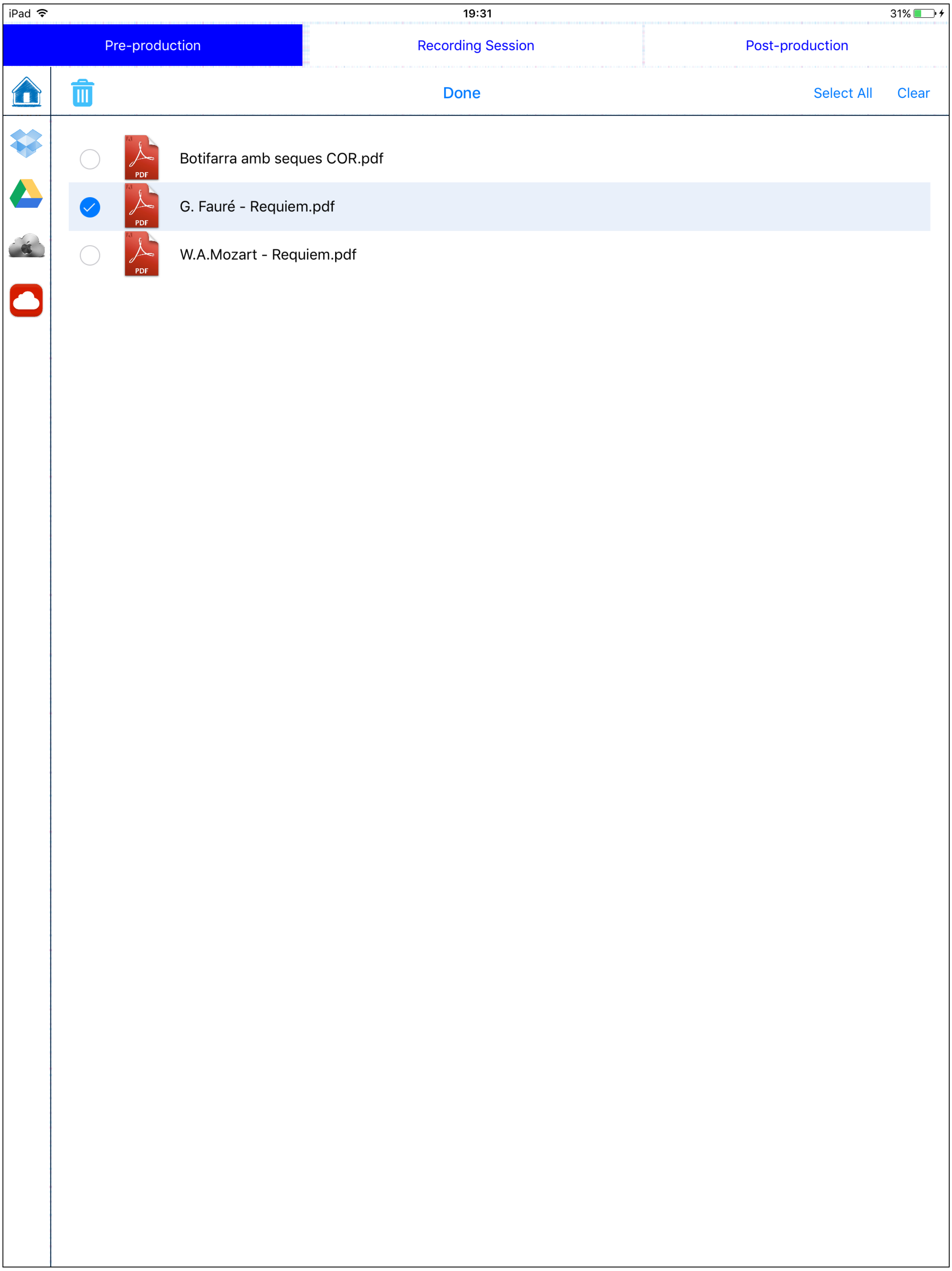
```
func updateNote (_ n:Int)
{

    let appDel: AppDelegate = UIApplication.shared.delegate as! AppDelegate
    let context: NSManagedObjectContext = appDel.managedObjectContext
    let request = NSFetchRequest<NSFetchRequestResult>(entityName: "Anotacions")
    let documentPredicate = NSPredicate(format: "document = %@",
        arxiusPdfs[pdfSel].nom) //Posa com a condició que el document sigui
        igual a l'arxiu que he triat
    let paginaPredicate = NSPredicate(format: "pagina = %@", String(docNotes[n].
        pag))
    let idPredicate = NSPredicate(format: "id = %@", docNotes[n].id)
    request.returnsObjectsAsFaults = false

    let andPredicate = NSCompoundPredicate(type:
        NSCompoundPredicate.LogicalType.and, subpredicates: [documentPredicate,
        paginaPredicate, idPredicate])

    request.predicate = andPredicate

    do
    {
        let results = try context.fetch(request)

        if (results.count>0)
        {
            let managedObject = results[0]
            (managedObject as AnyObject).setValue(docNotes[n].nota, forKey:
                "nota")
            (managedObject as AnyObject).setValue(docNotes[n].sistema, forKey:
                "sistema")
            //(managedObject as AnyObject).setValue(docNotes[n].take, forKey:
                "take")
            (managedObject as AnyObject).setValue(docNotes[n].x, forKey: "x")
            (managedObject as AnyObject).setValue(docNotes[n].y, forKey: "y")

            do
            {
                try context.save()
                print("desat")
            }
            catch
            {

            }

        }
    }
    catch
    {
        print("fail")
    }
}


//// Funció per eliminar una entrada de la base de dades. Necessita entrar
    l'n de la nota en qüestió
func deleteNote (_ n:Int)
{

    docNotes[n].visible=false
```

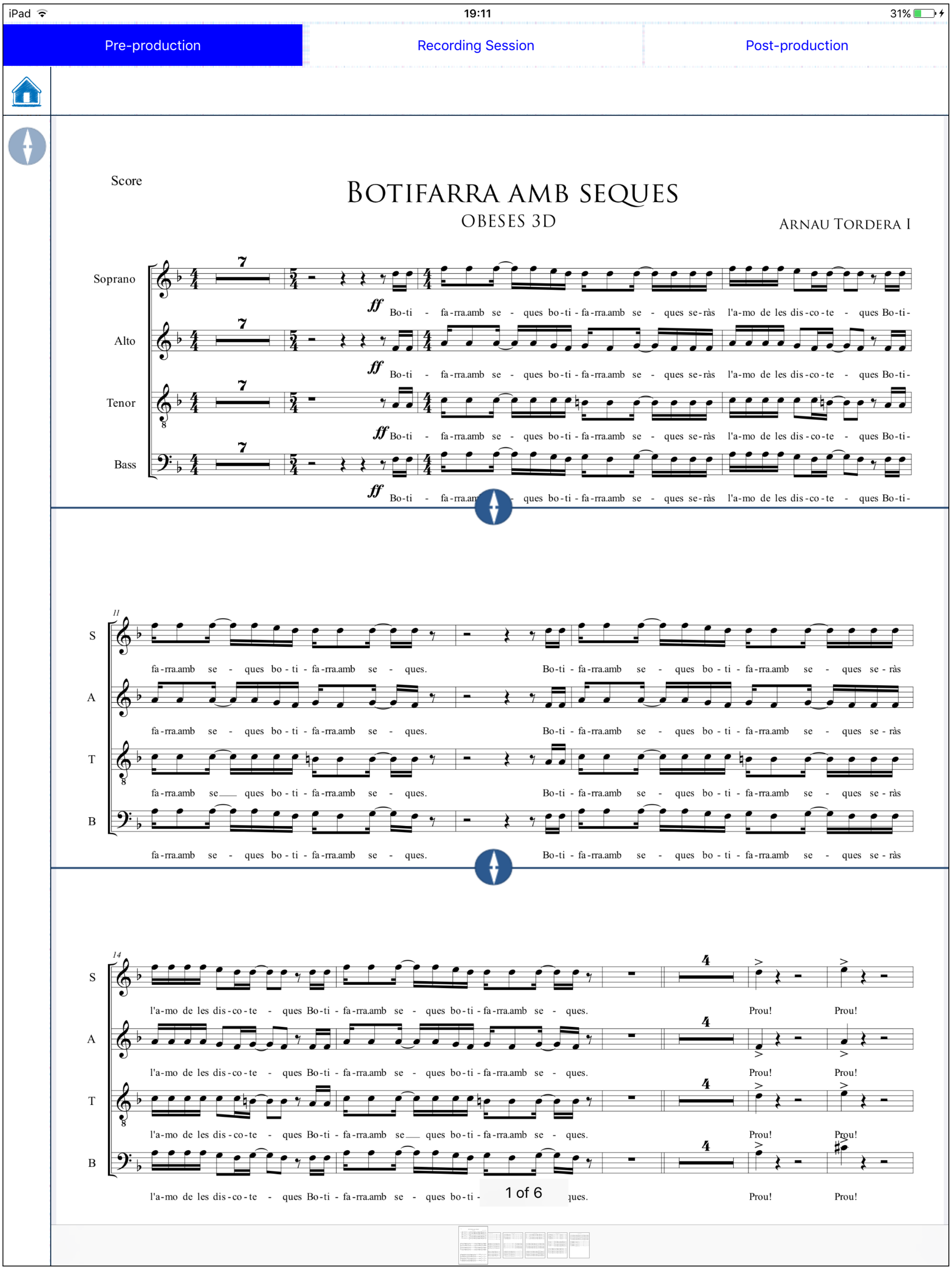
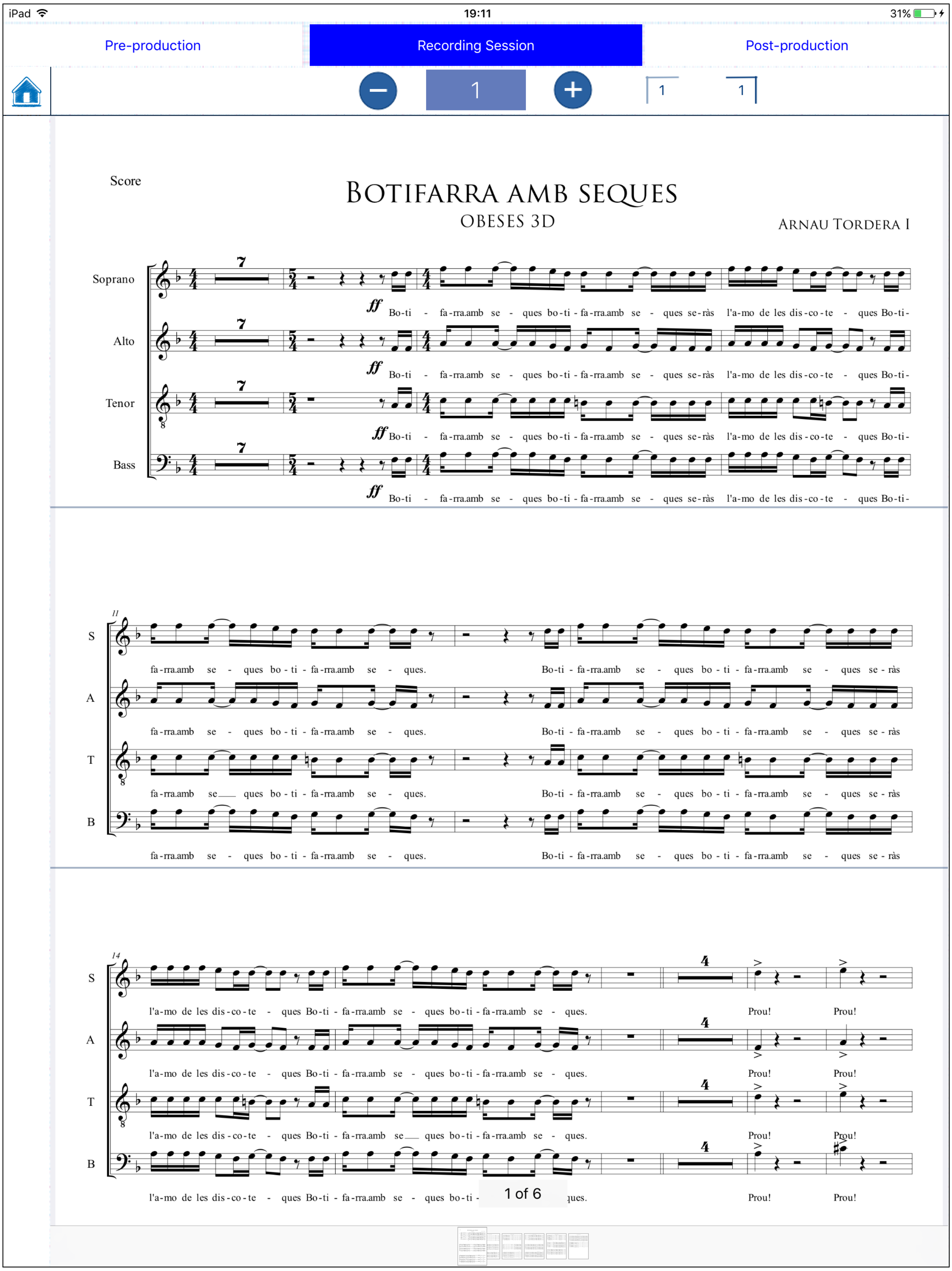
```
        docNotes[n].boto.removeFromSuperview()

        let appDel: AppDelegate = UIApplication.shared.delegate as! AppDelegate
        let context: NSManagedObjectContext = appDel.managedObjectContext
        let request = NSFetchRequest<NSFetchRequestResult>(entityName: "Anotacions")
        let documentPredicate = NSPredicate(format: "document = %@",
            arxiusPdfs[pdfSel].nom) //Posa com a condició que el document sigui
            igual a l'arxiu que he triat
        let notaPredicate = NSPredicate(format: "nota = %@", String(docNotes[n].nota))
        let paginaPredicate = NSPredicate(format: "pagina = %@", String(docNotes[n].
            pag))
        let takePredicate = NSPredicate(format: "take = %@", String(docNotes[n].take))
        let idPredicate = NSPredicate(format: "id = %@", docNotes[n].id)
        request.returnsObjectsAsFaults = false

        let andPredicate = NSCompoundPredicate(type:
            NSCompoundPredicate.LogicalType.and, subpredicates: [documentPredicate,
            notaPredicate, paginaPredicate, takePredicate, idPredicate])

        request.predicate = andPredicate

        do
        {
            let results = try context.fetch(request)

            if (results.count>0)
            {
                for result in results as! [NSManagedObject]
                {

                    context.delete(result)

                    do
                    {
                        try context.save()
                        print("eliminat")
                    }
                    catch
                    {

                    }

                }
            }
        }
        catch
        {
            print("No s'ha eliminat bé")
        }
    }
```

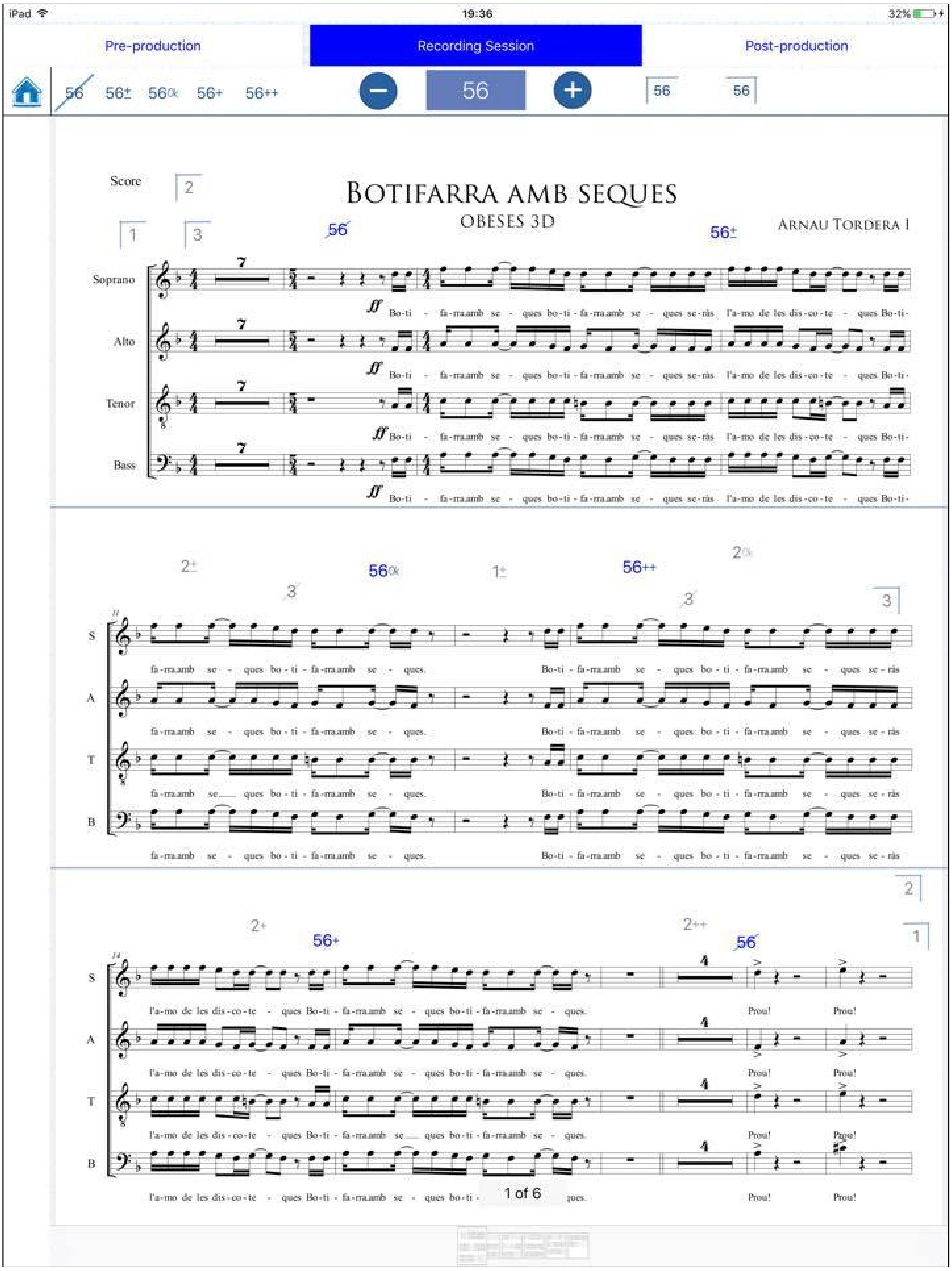
A.5 Captures del prototip




iPad
19:10
31%
Pre-production
Recording Session
Post-production
LogIn Dropbox
LogOut Dropbox
PDF
Botifarra amb seques COR.pdf


iPad
19:31
31%
Pre-production
Recording Session
Post-production
Done
Select All
Clear
Botifarra amb seques COR.pdf
G. Fauré - Requiem.pdf
W.A.Mozart - Requiem.pdf


iPad
19:11
31%
Pre-production
Recording Session
Post-production
Score
BOTIFARRA AMB SEQUES
OBESES 3D
ARNAU TORDERA I
Soprano
Alto
Tenor
Bass
Bo-ti - fa-rra.amb se - ques bo-ti - fa-rra.amb se - ques se-ràs l'a-mo de les dis-co-te - ques Bo-ti-
fa-rra.amb se - ques bo - ti - fa-rra.amb se - ques. Bo-ti - fa-rra.amb se - ques bo - ti - fa-rra.amb se - ques se - ràs
l'a-mo de les dis-co-te - ques Bo-ti - fa-rra.amb se - ques bo-ti - fa-rra.amb se - ques. Prou! Prou!
1 of 6


iPad
19:11
31%
Pre-production
Recording Session
Post-production
1
1
1
Score
BOTIFARRA AMB SEQUES
OBESES 3D
ARNAU TORDERA I
Soprano
Alto
Tenor
Bass
ff
Bo-ti - fa-rra.amb se - ques bo-ti - fa-rra.amb se - ques se-ràs l'a-mo de les dis-co-te - ques Bo-ti-
S
A
T
B
fa-rra.amb se - ques bo - ti - fa-rra.amb se - ques. Bo-ti - fa-rra.amb se - ques bo - ti - fa-rra.amb se - ques se - ràs
l'a-mo de les dis-co-te - ques Bo-ti - fa-rra.amb se - ques bo-ti - fa-rra.amb se - ques. Prou! Prou!
1 of 6


iPad
19:36
32%
Pre-production
Recording Session
Post-production
56
Score
BOTIFARRA AMB SEQUES
OBESES 3D
ARNAU TORDERA I
Soprano
Alto
Tenor
Bass
Prou!
1 of 6

Bibliografia

Hepworth-Sawyer, R. and Golding, C. (2011).
What is music production?
Focal Press.

Hillegas, A. and Ward, M. (2013).
Objective-C Programming.
Big nerd ranch.

King, R. (2017).
Recording Orchestra and Other Classical Music Ensembles.
Focal Press.

Martin, R. C. (2009).
Clean Code, A Handbook of Agile Software Craftsmanship.
Prentice Hall.

Monefa, N. (2015).
The Ultimate Beginers guide from app programming and development.
Autoedició - AmazonBooks.