MusicAlignApp. Creació d'una aplicació per alinear partitures musicals

Oscar Pozo Fernández

Resum— Aquest projecte neix amb l'objectiu de facilitar la creació de dades etiquetades per entrenar sistemes de reconeixement òptic de partitures musicals (OMR). Per aquest motiu, es proposa una aplicació interactiva i tàctil que permetrà als usuaris realitzar l'alineament de les partitures digitalitzades d'una forma més ràpida, intuïtiva i precisa, a causa del seu enfocament tàctil. La proposta d'aplicació permet als usuaris carregar la imatge processada per l'OMR, i retallar-la en els diferents sistemes de la seva partitura, que coincideixen amb els diversos fitxers MusicXML resultants. Un cop carregat, es renderitzen aquests fitxers, juntament amb la imatge retallada que li correspon a cada un d'ells, i es mostra a l'usuari en format SVG, per tal que aquest tingui una representació visual de la partitura, i pugui realitzar un alineament precís dels elements que es troben en la imatge retallada del manuscrit, dibuixant a la imatge manuscrita, amb el llapis "*Stylus Pen*", els elements que corresponen amb la traducció del fitxer editable.

Paraules clau— OMR, Alineament, Verovio, Kotlin, Firebase, Clean Architecture, Creació dades etiquetades

Abstract— This project was initiated with the aim of facilitating the creation of ground-truth data for training optical music recognition (OMR) systems. For this reason, an interactive and tactile application is proposed, allowing users to align digitized scores in a faster, more intuitive, and accurate way due to its tactile approach. The proposed application enables users to upload the image processed by OMR and crop it into different systems of their score, which correspond to the various resulting MusicXML files. Once uploaded, these files are rendered, along with the cropped image corresponding to each of them, and displayed to the user in SVG format, providing a visual representation of the score. This allows for precise alignment of the elements found in the cropped image of the manuscript, drawing the corresponding elements from the editable file translation on the manuscript image with the "Stylus" pen.

Index Terms— OMR, Alignment, Verovio, Kotlin, Firebase, Clean Architecture, Ground-truth

1 Introducció

El reconeixement òptic de partitures musicals (OMR) [1] consisteix a reconèixer i interpretar imatges de partitures musicals i convertir-les a format editable, com per exemple MIDI o MusicXML [2], és a dir, un OCR musical.

Si bé l'OMR ha experimentat un gran avenç en els últims anys, amb aplicacions de software comercial per a partitures impreses, encara té limitacions, sobretot per reconèixer partitures manuscrites o bé capturades amb dispositius mòbils. De fet, l'OMR encara comet molts errors en interpretar la notació musical, especialment en partitures manuscrites amb cal·ligrafia complexa o deteriorada. A més, hi ha una manca de dades etiquetades de partitures manuscrites per entrenar sistemes de OMR, el que dificulta l'ús d'arquitectures d'aprenentatge profund (deep learning).

Per aquest motiu es proposa una aplicació interactiva i tàctil que permet als usuaris alinear imatges de partitures digitalitzades amb el corresponent format MusicXML d'una manera més ràpida, intuïtiva i precisa, millorant significativament l'experiència d'usuari i agilitzant el procés de creació de dades d'entrenament.

- *E-mail de contacte: o.p.fernandez4@gmail.com*
- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: Alicia Fornés Bisquerra i Pau Torras Coloma (Centre de Visió per Computador)*
- *Curs 2023/24*

2 Estat de l'art

En el context de sistemes de OMR, trobem diferents treballs relacionats. En el treball presentat per Riba, Fornés i Lladós (2016) [3] es proposa un mètode per comparar partitures musicals que facilita la identificació de variacions entre elles. Torras, et al. (2021) [4] exploren la integració de models de llenguatge en arquitectures Seq2Seq per a la millora del reconeixement de partitures manuscrites històriques, aconseguint resultats satisfactoris. Finalment, Baró, Riba i Fornés (2022) [5] suggereixen l'ús de xarxes neuronals gràfiques per al reconeixement de partitures musicals, aprofitant la seva capacitat per a manejar representacions n-dimensionals.

Per la part d'aplicacions que permeten treballar amb partitures musicals a partir d'un arxiu MusicXML o MEI, podem trobar *Mei-friend* [6] o *MuseScore* [7]. *Mei-friend* ens mostra el contingut del fitxer musical i la seva partitura renderitzada, permetent modificar el contingut del fitxer, mostrant aquestes correccions sobre la partitura, de forma instantània. *MuseScore,* en canvi, proporciona una interfície per modificar els elements de la partitura de forma interactiva, intuïtiva i ràpida, permetent la correcció, creació, reproducció i impressió en alta qualitat de partitures musicals. D'igual forma, té compatibilitat amb una àmplia gamma de fitxers musicals, entre ells, MusicXML.

Un cop realitzat l'estat de l'art, no es coneix cap aplicació per alinear partitures musicals, reforçant d'aquesta forma, la necessitat de realitzar aquest projecte.

3 Objectius

En aquest punt s'explicaran els diferents objectius del projecte:

1. Estudiar l'estat de l'art de OMR
2. Analitzar, Dissenyar i Desenvolupar una aplicació compatible per a dispositius *Android* amb el llapis tàctil *Stylus Pen,* que permeti fer alineaments de partitures musicals a partir de la imatge del manuscrit de la partitura original i el fitxer musical en format MusicXML
3. Representar imatges interactives en format SVG a partir de fitxers MusicXML.
4. Realitzar la gestió del servidor i dels usuaris.
5. Implementar l'aplicació seguint una metodologia Agile [8] desglossada en iteracions
6. Realitzar Software Testing sobre les diferents funcionalitats implementades al llarg del desenvolupament de l'aplicació.
7. Testejar l'aplicació amb usuaris finals
8. Emprar bones pràctiques de Software

4 Metodologia

Durant el projecte s'ha seguit una metodologia *Agile* desglossada en iteracions, amb l'objectiu de proporcionar flexibilitat i adaptabilitat durant el seu desenvolupament. D'aquesta forma, ha sigut possible realitzar una millora contínua i una resposta eficaç als canvis, igual que una millor col·laboració amb els s*takeholders*.

Així doncs, seguint els criteris marcats pels objectius de la metodologia esmentats anteriorment, s'ha prioritzat la flexibilitat i la comunicació amb els *stakeholders* i els usuaris finals, per permetre adaptacions segons les necessitats i els requisits de l'aplicació i per garantir una comprensió clara d'aquesta. A més, s'ha buscat fer entregues incrementals operatives, de forma que ha permès una avaluació continua i la possibilitat d'incorporar retroalimentació durant el procés.

Alguns exemples d'aplicació d'aquesta metodologia es reflecteixen a l'apartat d'anàlisi de requisits, on es podrà observar com s'han anat modificant al llarg de les diferents iteracions, amb l'objectiu de tenir una millor adaptació en l'ús final que se li donarà a l'aplicació.

5 Anàlisi de Requisits

A continuació es mostraran els Requisits Funcionals afegits al llarg de les iteracions del projecte, els Requisits Funcionals eliminats i els Requisits No Funcionals del projecte.

Respecte als Requisits Funcionals:

Iteració 1:

1. L'usuari ha de poder pujar i eliminar arxius MusicXML i imatges de la galeria del dispositiu.
2. L'usuari ha de poder crear, eliminar i visualitzar cada un dels projectes.
3. L'usuari ha de poder visualitzar la partitura renderitzada a partir del fitxer musical i la imatge del manuscrit corresponent.
4. L'aplicació ha de permetre l'usuari navegar entre els elements de la partitura de forma manual (element anterior i següent amb botons) i de forma automàtica (polsant *play* i *pause*)
5. L'usuari ha de poder marcar l'àrea de la imatge del manuscrit on es troba l'element que es vol alinear, amb el llapis tàctil *Stylus Pen*
6. En cas de navegar a través d'un element ja alineat, l'aplicació ha de mostrar la traçada del seu alineament a la partitura manuscrita, permetent a l'usuari eliminar-la i tornar a realitzar-la.
7. L'usuari ha de poder fer zoom i desplaçar en totes direccions la imatge de la partitura manuscrita, adaptant les traçades dels alineaments a aquests canvis.
8. L'usuari ha de poder guardar i recuperar el progrés de cada alineament.

Iteració 2:

9. L'aplicació ha de mostrar els elements SVG de la partitura de diferents colors, segons si han estat alineats o reconeguts.
10. L'SVG de la partitura ha de navegar de forma automàtica al següent element un cop l'usuari ha acabat de realitzar un alineament.
11. L'usuari ha de poder gestionar l'estat de la seva sessió (registrar-se, iniciar i tancar) a partir del seu

correu electrònic i contrasenya.
12. L'usuari ha de poder veure la data i hora de l'última modificació de cada projecte.

Iteració 3:
13. L'aplicació ha de permetre retallar la imatge de la partitura manuscrita a quan es crea un nou projecte.
14. L'aplicació ha de permetre fer l'alineament de cada sistema, amb la imatge retallada i l'SVG corresponent.
15. El color de les traçades dibuixades en realitzar l'alineament, ha de ser del mateix color que il·lumina els elements del SVG de la partitura.
16. L'aplicació ha de mostrar l'opció de guardar i finalitzar el projecte un cop es realitzi l'alineament de l'últim sistema de la partitura.
17. L'aplicació ha de mostrar el nom del sistema que s'està alineant.
18. Per defecte, l'aplicació ha de mostrar a la imatge del sistema, la traçada de l'alineament dels elements anteriors i posteriors de l'SVG, respecte a l'element que s'està alineant.
19. L'usuari ha de poder indicar el nombre d'elements anteriors i posteriors de l'SVG, dels quals es mostrarà la traçada del seu alineament.
20. L'aplicació ha de permetre amagar les traçades dels elements anteriors i posteriors de l'SVG.

Respecte als Requisits Funcionals eliminats del projecte:
1. L'aplicació ha de permetre l'opció de registrar-se i iniciar sessió com a administrador
2. L'usuari ha de poder carregar el fitxer MusicXML amb modificacions dins d'un projecte ja començat.

I per acabar, respecte als Requisits No Funcionals:
1. L'aplicació ha de ser compatible amb Tablets *Android*
2. L'aplicació ha de fer ús de diàlegs de confirmació com a resposta a diverses accions de l'usuari, on l'estat anterior és difícil de recuperar.
3. Només els usuaris que han iniciat sessió podran fer ús de l'aplicació.
4. L'aplicació ha de mostrar feedback a l'usuari en tot moment, a través de missatges d'error, barres de progrés o diàlegs.
5. L'aplicació ha de ser *user friendly.*
6. El sistema ha de ser fàcil de mantenir i actualitzar.

6 Planificació

La planificació que es va seguir per a la realització d'aquest projecte es va definir en el Diagrama de Gantt (Apèndix A1, Fig. 11) i va consistir en cinc fases que es veuran a continuació.

6.1 Inici, introducció i planificació

En aquesta primera fase es va fer una primera investigació sobre l'estat de l'art i les eines necessàries per al projecte, com l'estructura dels fitxers MusicXML, Verovio[9], i les tecnologies de desenvolupament d'aplicacions Android (Kotlin[10], Interfícies XML[11] i Firebase[12]). A més, es van identificar els objectius i es va realitzar la planificació del projecte.

6.2 Anàlisis de Requisits i Disseny

En aquest punt es van identificar i classificar els requisits funcionals per a la primera iteració del desenvolupament, així com els requisits no funcionals. Es van crear diagrames de casos d'ús, de seqüència i de classes, i es va dissenyar una primera versió de totes les pantalles de l'aplicació per definir l'estructura i funcionament d'aquesta.

6.3 Implementació

La fase d'implementació, planificada en 3 iteracions de 15 dies, seguint la metodologia *Agile* basada en iteracions, va servir per desenvolupar les funcionalitats identificades en la primera iteració i per fer reunions amb els *stakeholders* per assegurar que es complien les seves necessitats i per afegir nous requisits per a futures iteracions.

6.4 Test amb usuari final

Acabat el desenvolupament, es va fer un període de proves amb un expert en musicologia, usuari final de l'aplicació, per provar-la en un entorn real i donar feedback sobre errors i millores a implementar en futures versions.

6.4 Documentació final

En l'última de les fases, es va realitzar la documentació final del treball, que inclou redactar la proposta d'informe final, el dossier i la presentació *PowerPoint*.

7 Disseny

La fase de disseny va servir per acabar d'analitzar i definir el funcionament de l'aplicació, realitzar diagrames amb *PlantUML* [13] i dissenyar les pantalles d'aquesta. A continuació es mostrarà el diagrama de casos d'ús (Fig. 1), mentre que la resta de diagrames es trobaran a l'Apèndix A2.

7.1 Diagrama de Casos d'Ús i de Seqüència

Primer es va realitzar un diagrama de casos d'ús (Fig. 1) i un diagrama de seqüència (Apèndix A2, Fig. 12 i Fig. 13) per definir el flux complet de l'aplicació, des que l'usuari es registra, fins que fa la validació de la partitura i guarda el seu progrés.

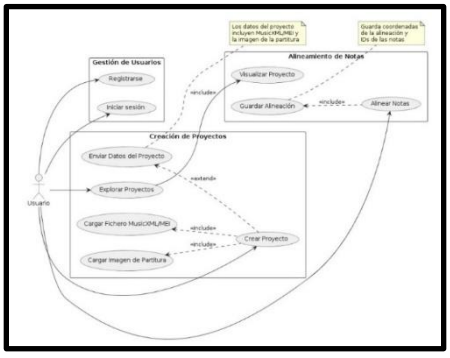

Fig. 1. Diagrama de casos d'ús de l'aplicació

7.2. Diagrames de Classes

Un cop realitzat els diagrames de casos d'ús i de seqüència, es va passar a dissenyar diagrames de classes UML. L'aplicació s'ha implementat seguint bones pràctiques de Software, emprant MVVM[14], *Clean Architecture*[15] i Injecció de dependències amb Dagger-Hilt[16], entre d'altres. Per aquest motiu, les classes del codi estan molt desacoblades i la creació d'aquestes és molt extensa, per tant, es va decidir realitzar diagrames de classes UML per cada una de les funcionalitats principals: *login*, registre, *home*, creació de projectes i validació de partitures, inclús havent de realitzar varis en el cas de la creació d'un projecte, degut a la complexitat i extensió de la funcionalitat (Apèndix A2, Fig. 14, Fig. 15, Fig. 16, Fig. 17, Fig. 18, Fig. 19 i Fig. 20).

7.3 Pantalles de l'aplicació

Com a última part del disseny, es va fer una primera versió de les diferents pantalles de l'aplicació (Apèndix A2, Fig. 21, Fig. 22, Fig. 23 i Fig. 24), que van en correspondència amb els anteriors diagrames de classes. D'igual forma, aquestes van patir modificacions amb el pas de les iteracions per adaptar-se als nous requisits que s'afegien, fins a arribar al disseny final, on inclús es va haver de crear una nova pantalla per retallar les imatges (Fig. 2, Fig. 3, Fig. 4, Fig. 5 i Fig. 6).

8 IMPLEMENTACIÓ

Respecte a la implementació del projecte, es poden diferenciar dos grans blocs: el servidor, i l'aplicació. Dins de l'aplicació va haver-hi parts principals: d'una banda, el *front-end* i la lògica implementada amb *Kotlin*, que defineix tota la interfície d'usuari i la funcionalitat de l'aplicació; d'altra banda, el codi web desenvolupat amb *JavaScript* [17], que utilitza *Verovio* per renderitzar els fitxers MusicXML en SVG i permet realitzar modificacions de cada element per millorar l'experiència d'usuari. Aquest codi web s'utilitzarà en un *WebView*.

8.1 Servidor

El servidor del projecte es va desenvolupar amb *Firebase*. A continuació, es detallaran les eines utilitzades i la seva utilització.

8.1.1. Firebase Authentication

La primera de les eines usades va ser *Firebase Authentication* [18], que va permetre realitzar la gestió d'usuaris de forma correcta i segura. Alguna de les funcionalitats que es van fer servir són:

1. **Registre d'usuaris** amb el seu correu electrònic i una contrasenya.
2. **Inici de sessió** per part dels usuaris que ja han estat registrats.
3. **Gestió de sessions** de forma segura, portant el control dels *id tokens* o *refresh tokens.*

8.1.2. Firebase Storage

La segona eina que es va utilitzar, va ser *Firebase Storage* [19], un servei d'emmagatzematge al núvol que es fa servir per guardar i pujar contingut generat per l'usuari. En aquest cas, es va utilitzar per guardar les imatges i els fitxers MusicXML corresponent a cada partitura, juntament amb els fitxers JSON [20] que recullen tota la informació relacionada amb la validació del sistema corresponent: l'*id* de l'element alineat i les seves coordenades, l'últim element alineat, o l'últim element sobre el qual ha parat l'usuari, amb l'objectiu de retornar al mateix estat d'alineament un cop s'obri el projecte quan s'hagi tancat.

8.1.3. Firestore Database

L'última eina que es va fer servir és *Firestore Database* [21]. Un servei de base de dades *NoSQL* en temps real basada en documents, que en el cas d'aquesta aplicació, es va fer utilitzar per emmagatzemar dades relacionades amb cada un dels projectes. Dins d'aquesta estructura de dades tenim atributs com:

1. ***currentSystem***: Indica en quin sistema de la partitura està treballant l'usuari.
2. ***isFinished***: Indica si el projecte ha estat finalitzat o no.
3. ***lastModified***: Emmagatzema la data i hora de l'última modificació del projecte.
4. ***numSystems***: Indica el nombre de sistemes que té la partitura a validar.
5. ***originalImageUrl***: Conté l'URL de la imatge amb la partitura manuscrita completa.
6. ***projectName***: Conté el nombre del projecte.

A part d'aquests atributs, cada un dels documents dels sistemes té 3 col·leccions (*files*, *images* i *json*), els quals contenen el nom i els *url* dels diferents fitxers, imatges i *JSONs*.

8.2 Front-End i Lògica

L'aplicació s'ha desenvolupat en *Android* natiu, fent ús del llenguatge *Kotlin* per a la seva lògica i del sistema de vistes *XML* per a la interfície d'usuari. Dins d'aquesta implementació, i com ja s'ha comentat al llarg de l'article, s'ha seguit *MVVM*, *Clean Architecture* i injecció de dependències.

8.3 Webview i Verovio

El tercer gran bloc del projecte va ser la integració d'un *Webview* per a renderitzar les partitures musicals que venen donades pels arxius MusicXML, i convertir-les en SVG. Dins d'aquest, les funcionalitats per poder manipular els diferents elements de la partitura processada per Verovio van estar implementades a partir de codi *Javascript*, permetent agregar identificadors als elements o canviar el seu color segons el seu estat de processament. Aquestes modificacions van permetre una interacció visual i dinàmica, que millora en gran mesura l'experiència de l'usuari treballant amb aquestes partitures.

8.3.1. Comunicació Aplicació - WebView

Una altra part important de la implementació va ser de quina forma es va aconseguir la comunicació entre els components de l'aplicació i el codi del *WebView*, ja que aquesta comunicació és essencial per permetre interaccions dinàmiques i en temps real des de l'aplicació cap a l'SVG de la partitura.

En aquest cas, la comunicació entre aplicació i *WebView* és més complexa que entre dos components interns de l'aplicació, com ara els elements dins una mateixa pantalla, o inclús comunicació entre elements que es troben en dues pantalles diferents. Això és degut al fet que el codi del *WebView* no és codi intern de l'aplicació nativa, sinó codi web extern injectat dins d'aquesta.
Aquesta comunicació és bidireccional, ja que és necessari tant comunicar-se des de l'aplicació fins al *WebView*, com a la inversa. Alguns exemples poden ser:

1. **Comunicació Aplicació - *WebView*:** Clicar un botó des de la interfície de l'aplicació i realitzar un canvi en l'SVG de la partitura generada per *Verovio* al *WebView*. Un d'aquests pot ser saltar al següent element de la partitura, canviant també el color de l'element anterior i l'actual.
2. **Comunicació *WebView* - Aplicació:** Enviar l'*id* de l'element de la partitura que s'alinearà cap a una pantalla de l'aplicació, o enviar el nombre màxim d'elements que conté aquell arxiu MusicXML.

Per fer possible aquesta comunicació, es va haver de crear una interfície de *JavaScript*, que permet, per una banda, que el codi *JavaScript* del *WebView* faci crides a mètodes de l'aplicació *Android* fets en *Kotlin*, i per una altra, permet definir mètodes dins d'aquesta que reaccionen a esdeveniments del *WebView*, i d'aquesta forma, utilitzant *StateFlows*, poden actualitzar l'estat de la pantalla, enviar dades o realitzar accions de forma immediata i reactiva.

9 Resultats

En aquest punt, presentarem els diferents resultats obtinguts en cada una de les parts de l'aplicació.

9.1 Login

La pantalla de *login* (Fig. 2) és la primera que es mostra quan l'usuari obre l'aplicació per primer cop. En aquesta, l'usuari que ja té un compte creat pot accedir al seu espai a través d'introduir el seu correu electrònic i la seva contrasenya. En cas de no estar registrat, la pantalla mostra una opció per navegar al registre. A més, aquesta inclou detalls que milloren l'experiència d'usuari, com ara, missatges d'error explicatius en casos com introduir un format de correu electrònic incorrecte, o una icona que permet a l'usuari veure el contingut de la contrasenya que està introduint.

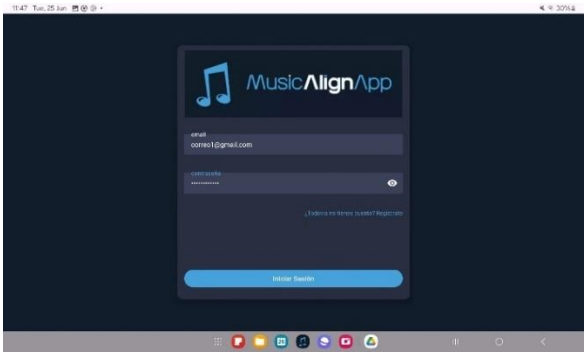

Fig. 2. Pantalla *login* on s'ha introduït correu electrònic i contrasenya.

9.2 Registre

La següent pantalla de registre (Fig. 3) conté el formulari que permet a l'usuari registrar-se a l'aplicació a partir del seu correu electrònic i la seva contrasenya. Per a realitzar aquestes accions, l'aplicació comprova que el format de les dades sigui el correcte, és a dir, que el correu introduït i la contrasenya són vàlides o que les dues contrasenyes introduïdes siguin exactament les mateixes. Aquesta pantalla dona l'opció de navegar de nou al *login* en cas que l'usuari ja estigui registrat, igual que veure el contingut del què s'està escrivint a les contrasenyes a partir de clicar una icona.

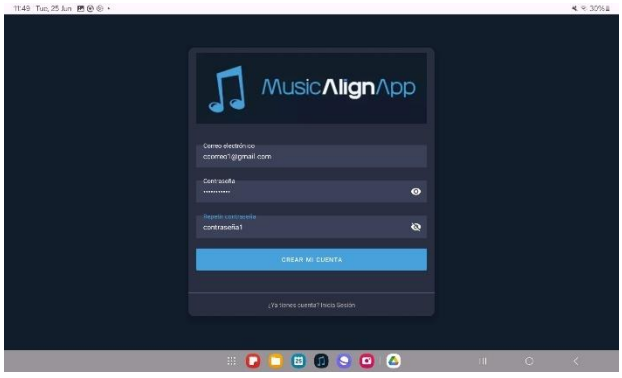

Fig. 3. Pantalla de Registre on s'ha introduït el correu electrònic i dos cops la contrasenya. Una de les contrasenyes és visible gràcies a la icona introduïda al camp del formulari.

9.3 Home

La *home* (Fig. 4) és la pantalla principal de l'aplicació, i la responsable de mostrar a l'usuari els projectes que ha creat, separant-los en 2 seccions. Per una part, es mostren els projectes que encara estan en curs, i per altre, es mostren els projectes que ja s'han marcat com a finalitzats. Referent a tots els projectes, es mostra la imatge original de la partitura manuscrita, el nom del projecte, la data i hora de l'última modificació i una icona d'una creu que permet eliminar el projecte. Donat que aquesta acció és delicada, abans d'esborrar-ho, l'aplicació mostra a l'usuari un diàleg de confirmació que evita realitzar aquesta acció de forma involuntària. Altres accions implementades en aquesta pantalla són el *logout*, que tanca la sessió de l'usuari i li envia a la pantalla de *login*, un *FloatingActionButton* que permet navegar a la pantalla de creació d'un projecte, o navegar a la pantalla d'alineament del projecte, clicant sobre ell.

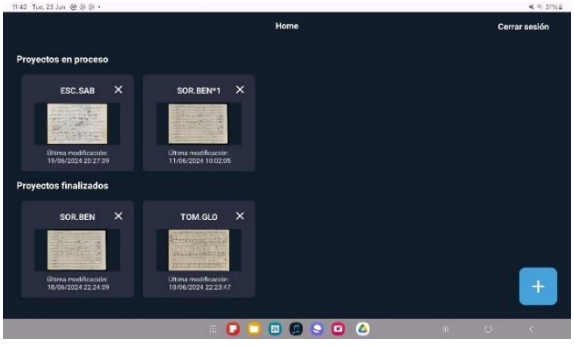

Fig. 4. Pantalla *home* amb projectes en curs i finalitzats

9.4 Crear projecte

La següent pantalla de l'aplicació (Fig. 5) és la encarregada de poder crear un projecte.

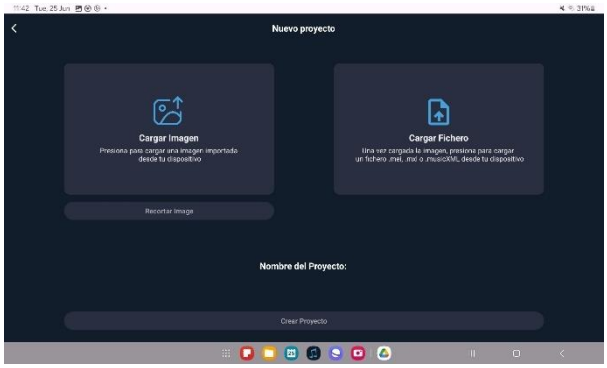

Fig. 5. Pantalla de creació de projectes. En aquest cas encara no s'ha carregat cap imatge ni fitxer per a la creació del projecte.

Per fer això, hi ha tres aspectes necessaris, la càrrega d'imatges, la selecció de fitxers i l'assignació de nom del projecte.

9.4.1 Selecció d'imatges

Quan l'usuari vol carregar una imatge clica sobre la secció de la pantalla destinada a aquest fi, provocant que l'aplicació obri la galeria del dispositiu sense sortir de l'aplicació. Dins d'aquesta, l'aplicació només mostrarà les imatges, evitant així que pugui seleccionar qualsevol altre fitxer que pugui portar a errors en la pròpia aplicació. Un cop l'usuari escull la imatge, l'aplicació el redirigeix a la pantalla de creació de projectes, puja la imatge al servidor i mostra a l'usuari quina és la imatge que ha escollit, donant-li l'opció a esborrar-la en cas d'haver-se equivocat.
Un cop l'usuari té carregada la imatge, s'habilita un botó que permet fer retalls sobre aquesta, ja que en casos on la partitura manuscrita té més d'un sistema l'usuari podrà retallar-la separant cada un d'aquests, treballant amb ells de forma individual.

9.4.2 Selecció de fitxers

Per part de la selecció de fitxers, el procés és molt semblant a la selecció d'imatges. En aquest cas, no només ha de seleccionar un fitxer, sinó que com cada sistema de la partitura té un fitxer, hi ha casos on per una partitura existeixen múltiples fitxers. Per aquesta raó, l'usuari pot seleccionar tots aquests projectes de cop, sense haver d'accedir a la galeria més d'una vegada. Quan ja són seleccionats, torna a la pantalla de creació de projectes, es guarden tots els fitxers al servidor, i a mesura que es van guardant, es van mostrant els seus noms perquè l'usuari pugui verificar que es guarden correctament.

9.4.3 Nom del projecte

L'altre element necessari per crear el projecte és el nom d'aquest, el qual s'assigna de forma automàtica a partir del nom de la partitura manuscrita original, estalviant un pas a l'usuari i fent aquest procés de creació més ràpid.

9.5 Alineació de partitura

L'última de les pantalles de l'aplicació té totes les funcionalitats relacionades amb l'alineament de la partitura (Fig. 6).

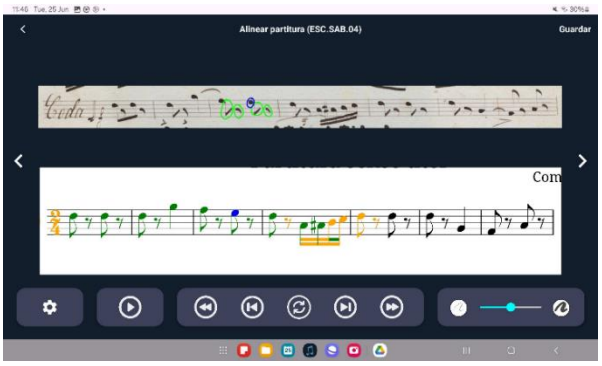

Fig. 6. Pantalla d'alineament de partitures.

Primerament, es farà un repàs pels diferents components que té la pantalla i el seu funcionament, per després fer una explicació de com és possible realitzar aquest exercici d'alineament amb les eines que hi ha a l'abast.

9.5.1 Imatge del sistema de la partitura manuscrita

El primer element és la imatge del sistema de la partitura manuscrita que va ser retallada a la creació del projecte (Fig. 7).

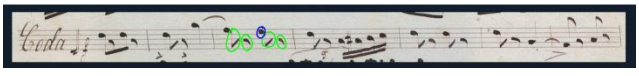

Fig. 7. Imatge del sistema retallat a partir de la partitura manuscrita. Es poden observar les traçades ja realitzades tant de l'element actual (blau) com dels dos elements anteriors i els dos elements posteriors (verd).

Aquesta imatge no només es carrega i queda fixa, sinó que es pot realitzar diferents accions sobre ella.

1. **Transformacions:** Es pot aplicar zoom sobre la imatge i realitzar desplaçaments sobre la mateixa cap a totes direccions.
2. **Dibuixos:** Utilitzant el *Pen Stylus* de la pròpia tauleta *Android*, es permet dibuixar traçades en qualsevol part de la imatge com si s'estigués dibuixant sobre el mateix full de la partitura manuscrita.

L'objectiu final d'aquestes transformacions a realitzar sobre la imatge és poder fer les traçades de l'alineament de cada element de forma més precisa, ja que hi ha molts elements que poden arribar a ser molt petits, i d'aquesta forma l'usuari pot adaptar la posició i zoom per poder realitzar aquesta traçada de forma precisa en cada un dels elements.

9.5.2 Sistema renderitzat en format SVG

Sota la imatge tenim el *WebView* (Fig. 7), que carrega la imatge de la partitura renderitzada per Verovio en format SVG, de tal forma que tenim la imatge del sistema de la

partitura manuscrita i el seu equivalent en format SVG.

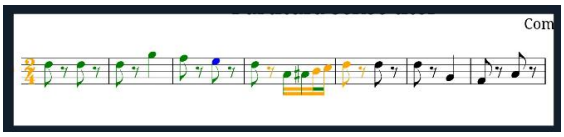

Fig. 8. WebView amb l'SVG d'un sistema d'una partitura musical a mig alineament. Es poden observar l'element actual, elements ja alineats, elements no alineats i elements no recorreguts.

Amb aquest, es podrà recórrer cada un dels elements de la partitura, i modificar el seu color de forma individual segons el seu estat de processament:

1. **Blau:** Indica l'element actual per alinear
2. **Verd:** Indica els elements que ja han estat alineats
3. **Taronja:** Indica els elements pels quals l'usuari ha accedit, però no ha realitzat el seu alineament.
4. **Negre:** Indica els elements que encara no han estat accedits per l'usuari.

9.5.3 Barra d'eines

A la part inferior es troba una barra d'eines amb diferents botons i funcionalitats (Fig. 9).

Fig. 9. Barra d'eines inferior situada a la pantalla d'alineament.

Per una banda, es troben 5 botons centrals que permeten fer una sèrie d'accions sobre l'SVG i les traçades d'alineament de la imatge manuscrita. A continuació s'explicarà cada un d'ells en ordre d'esquerra a dreta, segons la figura (Fig. 9).

1. Botó per tornar a l'últim element anterior que no hagi estat alineat.
2. Botó per tornar a l'element anterior immediat, independentment de si ja ha estat alineat o no .
3. Botó per tornar a alinear un element. Quan l'usuari està a un element que ja ha estat alineat, pot desfer la traçada ja feta i realitzar-ne una de nova.
4. Botó per avançar al següent element immediat, independentment de si ja ha estat alineat o no .
5. Botó per avançar al següent element que no estigui alineat.

A l'esquerra d'aquests 5 botons, es trobarà un altre que permet activar el mode de navegació automàtic sobre els elements de l'SVG. Primerament, aquest botó és presenta en la seva forma de *Play*, que al ser clicat, el sistema anirà avançant en ordre per cada un dels elements, mostrant de color blau l'element actual en cada pas que avanci, i deixant del mateix color els elements que ja ha recorregut. D'altra banda, un cop s'ha activat aquest mode automàtic, el mateix botó de *Play* passa a estar en el mode *Stop*, el qual para la navegació automàtica i retorna al mode manual, on es tornaran a utilitzar els botons per navegar sobre l'SVG. El següent element que es troba a la barra d'eines, és el col·locat a la dreta dels 5 botons. En aquest cas, veiem un *Slider* que permet a l'usuari modificar el gruix de les traçades fetes sobre la imatge de la partitura manuscrita, com també que les traçades que es realitzaran a continuació. El problema que soluciona aquesta eina és que hi ha molts tipus diferents de partitures manuscrites que necessiten més o menys precisió depenent de la mida dels seus elements, per exemple:

1. **Partitura musical amb molts elements**. Els elements estan molt a prop un dels altres, per tant, es necessita un traçat més fi per obtenir més precisió.
2. **Partitura musical amb pocs elements.** Permeten un traçat més gruixut, que serà més còmode per l'usuari a l'hora de realitzar la línia i de veure com ha quedat.

Per acabar, l'altre element de la barra d'eines és el botó que accedeix a la configuració de l'alineament. Aquesta és la secció de més a l'esquerra de la barra d'eines i conté opcions addicionals que milloren l'experiència d'usuari en l'alineament i es veuran més en detall a l'apartat de configuració de l'alineament.

9.5.4 Configuració d'alineament

Com ja s'ha mencionat, aquesta secció proporciona opcions addicionals sobre l'alineament i, en clicar-la, l'aplicació obre un diàleg que mostra tres elements (Fig. 10).

Fig. 10. Diàleg de configuració d'alineament. La *Checkbox* es troba clicada i per tant, els botons del nombre de traçats es troben des habilitats.

A continuació s'esmentaran i explicaran cada un d'aquests en ordre d'aparició de dalt a baix del diàleg.

1. **Botó per mostrar més informació útil sobre la pantalla d'alineament**. L'objectiu és mostrar informació visual sobre els botons per navegar pels elements de l'SVG i sobre el significat dels colors que prenen els elements del mateix (Fig. 11).

Fig. 11. Diàlegs d'informació sobre botons i colors de l'alineament.

2. ***Checkbox* per deixar de mostrar les traçades dels elements que es troben al voltant de l'element**

actual. Amb aquesta opció l'usuari pot decidir mostrar o amagar totes les traçades que no siguin de l'element actual.

3. **Comptador per escollir el nombre de traçades a mostrar al voltant de l'element actual**. A diferència que en el punt 2, aquesta opció permet definir el número de traçades que es mostraran a la imatge manuscrita.

Per tenir una idea més clara dels punts 2 i 3, a les (Fig. 12, Fig. 13, Fig.14 i Fig.15) es pot veure un exemple de com queda el diàleg i la pantalla d'alineament en diferents escenaris segons la configuració escollida.

Fig. 12. Diàleg de configuració amb l'opció d'amagar les traçades que no són de l'element actual habilitada. Es pot observar que encara que el número de traçades que s'indica és 4, aquesta opció està des habilitada.

Fig. 13. Pantalla d'alineament amb la configuració de la (Fig. 12). Es pot observar que només mostra la traçada de l'element actual (blau).

Fig. 14. Diàleg de configuració amb l'opció d'amagar les traçades que no són de l'element actual des habilitada i el número de traçades a mostrar a 4. Es pot observar que ara els botons per escollir el nombre de traçades a mostrar està habilitat.

Fig. 15.Pantalla d'alineament amb la configuració de la (Fig. 14). Es pot observar que es mostra la traçada de l'element actual (blau), i les traçades dels 4 elements anteriors i posteriors (verd).

9.5.5 Navegació entre sistemes

El següent element important es tracta de les fletxes que permeten saltar de sistema en sistema dins d'un mateix projecte. Aquestes es troben a l'esquerra i la dreta de la imatge i SVG de cada sistema (Fig. 6), on clicant a cada fletxa es navega al sistema anterior o posterior respectivament. A més, en cas de trobar-se en el primer element, la fletxa que permet navegar a l'anterior sistema no es mostra, d'igual forma que quan es troba a l'últim sistema, no es mostra la fletxa que permet navegar al següent sistema.

9.5.6 Tancament i guardat de l'alineament

Per acabar, aquesta pantalla (Fig. 6) conté diverses opcions per a tancar i guardar el progrés de validació de la partitura. Per una banda, l'usuari pot guardar el progrés en qualsevol moment amb l'opció *guardar* que es troba a la part superior dreta, però en cas que l'usuari es trobi en l'últim sistema de la partitura, l'usuari rebrà l'opció de guardar i finalitzar. Per altra banda, l'aplicació mostra diàlegs de confirmació de tancament quan l'usuari vol sortir de la validació o vol canviar de sistema, ja que amb aquestes dues accions el progrés es perdria en cas de no haver guardat. Per part del tancament, la pantalla també dona l'opció de sortir de la validació clicant *cerrar* o amb la fletxa de darrere de la pròpia tauleta *Android*.

10. TEST

10.1. Tests Unitaris i d'Integració

La idea principal del *testing* era automatitzar tests unitaris per a cada funcionalitat desenvolupada, però finalment, per qüestions de temps no es va poder completar amb èxit completament. Encara que a l'acabar de desenvolupar cada funcionalitat es provava per verificar el seu bon funcionament, no es van poder automatitzar tots els *tests* per tenir un bon control del funcionament global de l'aplicació. Per aquesta raó, va aparèixer algun error no esperat en funcionalitats que ja s'havien donat per acabades i que van suposar un impacte negatiu en el temps requerit per desenvolupar el projecte.

10.2. Test amb Usuari Final

Durant l'última part del projecte, es va comptar amb l'ajut d'un musicòleg que va accedir a fer ús de l'aplicació i

proporcionar feedback amb possibles millores i alguns errors de l'aplicació. Començarem comentant els errors generals de l'aplicació, quins van ser i de quina forma es van solucionar:

1. No guarda l'alineació de l'últim símbol del fitxer MusicXML quan arriba al final del sistema, sinó que l'últim element queda de color blau i per molt que es marqui a la imatge, no es guarda com alineat.
2. L'SVG de la partitura no ofereix per marcar lligadures de prolongació, calderons, tresets, trèmolos, símbols de repetició de compàs.

Respecte el punt 1, l'error es produïa perquè l'aplicació detectava que aquest era l'últim element i per tant evitava que avancés. En aquest cas, si no avança d'element, no es guarda l'alineament de l'anterior i, per tant, la solució va ser deixar que avancés com si hi hagués un altre element, i controlar aquest cas específic per evitar errors. I respecte al punt 2, es va parlar amb l'expert en musicologia per tal que preparés una partitura de prova que contingués tots aquests elements mencionats. Un cop es va tenir accés a aquesta partitura es va realitzar l'adaptació de cada un d'ells fins a aconseguir oferir l'alineament de cada un d'ells.

Un cop vistos els errors generals, mencionarem un parell de situacions més específiques on l'usuari es va trobar amb errors, comentant també quines mesures es van prendre.

1. En pitjar guardar, els fitxers XML fets fins llavors dins del projecte es van quedar bloquejats i només deixava editar els arxius que no s'havien processat.
2. En un punt de l'alineament, es va bloquejar l'opció de deixar de mostrar traçades... i l'aplicació s'ha tancat.

Respecte a aquests errors, com apareixen en situacions específiques que són difícils de replicar, s'ha introduït *Firebase Crashlytics* [22], una eina de gestió d'errors proporcionada per *Firebase* que permet tenir un control exhaustiu dels errors que apareixen a l'aplicació.

A banda d'aquests errors, també es van comentar possibles millores en diferents parts de l'aplicació.

1. En crear projectes amb partitures que tenen molts sistemes, és fàcil perdre's i no saber quin sistema estàs retallant.
2. A l'SVG, eliminar els elements "Instrument", "Títol" i "Compositor/Arranjador" del Verovio per tenir-ho més net.
3. A l'alineament, en sistemes múltiples, treballar per línies de forma horitzontal saltant compàs a compàs, en comptes de treballar per compassos de forma vertical.
4. Reiniciar la posició i zoom de la imatge al canviar de sistema, en comptes de mantenir el zoom i desplaçament de la imatge anterior.

11 Conclusions

Un cop finalitzat el projecte, podem dir que s'han aconseguit tots els objectius inicials, ja que s'han pogut representar imatges interactives en format SVG a partir de fitxers MusicXML fent ús de Verovio, i desenvolupar una aplicació que permet validar partitures musicals a partir de les seves imatges manuscrites originals i arxius generats per OMR. Per part de la implementació, s'ha seguit de forma satisfactòria una metodologia *Agile*, utilitzant *Kotlin* i sistema de vistes XML per implementar l'aplicació, i realitzant la gestió del servidor i dels usuaris amb *Firebase*. A més, tot el desenvolupament s'ha fet seguint bones pràctiques de software, amb una arquitectura MVVM ben definida, *Clean Architecture* i injecció de dependències per construir un codi desacoblat i ben estructurat. Per altra banda, s'ha aconseguit desenvolupar en el temps especificat, i no només s'han implementat la gran majoria dels requisits, a excepció d'alguns requisits de la iteració 3 que no hi va haver-hi temps a realitzar, sinó que s'han tractat detalls que milloren en gran mesura l'experiència d'usuari (Apèndix A3, Taula 4). I per últim, al final del projecte es va portar a terme una fase de test d'usuari, on es van descobrir errors que es van poder solucionar abans de l'entrega del treball, com també millores per a versions futures d'aquesta. L'únic punt negatiu sobre el desenvolupament del projecte ha estat el *testing*, ja explicat en l'apartat de tests unitaris i d'integració.

De cara al futur del projecte, hi ha intenció tant per part del professorat com de l'alumne a continuar amb el desenvolupament d'aquest projecte, realitzant noves versions on s'apliquin millores com les mencionades a l'apartat de test amb usuari final, i realitzar un manteniment de la mateixa, controlant errors que puguin anar apareixent amb el temps.

Agraïments

Per últim, voldria expressar el meu agraïment als meus tutors de projecte Alicia Fornés i Pau Torras, pel seu gran ajut i disponibilitat durant tot el projecte. La seva disposició per resoldre dubtes instantàniament ha estat crucial per l'avenç i èxit del projecte. A Carles Badal, per realitzar el test amb usuari final i proporcionar *feedback* que ha millorat significativament la qualitat de l'aplicació. Finalment, a la meva família i amics, per el seu gran suport emocional. El seu suport i ànims constants han sigut fonamentals per mantenir-me motivat i enfocat al llarg de tot el procés, no només d'aquest projecte, sinó de tot el transcurs de la carrera universitària. A tots vosaltres, moltes gràcies.

Bibliografia

[1] Hinchey, J. (2022, 25 novembre). *A review of Optical* music *recognition software*. Scoring Notes. https://www.scoringnotes.com/reviews/a-review-of-optical-music-recognition-software/

[2] *MusicXML for Exchanging Digital Sheet Music.* (2022, desebre 7). MusicXML. https://www.musicxml.com/

[3] *P. Riba, A. Fornés, J .Lladós.* **Towards the Alignment of Handwritten**

Music Scores. "Graphic Recognition. Current Trends and Challenges", Lecture Notes in Computer Science, vol. 9657, pp.103-116, B.Lamiroy and R.D.Lins (Eds). Editorial: Springer International Publishing, ISBN 978-3-319-52158-9, 2017. https://pages.cvc.uab.es/afornes/publi/chap_lncs/2016_LNCS_PRiba.pdf

[4] P.Torras, A.Baró, L.Kang, A.Fornés. **On the Integration of Language Models into Sequence to Sequence Architectures for Handwritten Music Recognition**. International Society for Music Information Retrieval Conference (ISMIR), 2021. https://pages.cvc.uab.es/afornes/publi/conferences/2021_ISMIR_PTorras.pdf

[5] *Arnau Baró, Pau Riba, and Alicia Fornés.* Musicgraph: Optical Music Recognition through Object Detection and Graph Neural Network. International Conference on Frontiers in Handwriting Recognition (ICFHR), 2022. https://pages.cvc.uab.es/abaro/papers/2022_ICFHR_ABaro.pdf

[6] *Mei-friend*. https://github.com/mei-friend/mei-friend

[7] *Software gratuito de composición y notación musical | MusesScore.* (s. f.-b). https://musescore.org/es

[8] Laoyan, S. (2024, 8 febrer). Agile Manifesto: qué son las metodologías agiles [2024] · Asana. *Asana.* https://asana.com/es/resources/agile-methodology

[9] *Verovio Reference book.* (s. f.). Reference Book For Verovio. https://book.verovio.org/

[10] *Kotlin Programming Language.* (s. f.). Kotlin. https://kotlinlang.org/

[11] *Diseños en Views*. (s.f.). Android Developers. https://developer.android.com/develop/ui/views/layout/declaring-layout?hl=es-419

[12] Firebase. (s.f.). *Firebase.* https://firebase.google.com/?hl=es

[13] PlantUML. (s.f.). Herramientas de código abierto que utiliza descripcions textuales simples para dibuixar diagrames UML. PlantUML.com. https://plantuml.com/es/

[14] Reyes, L. (2021, 7 desembre). Aplicando el patrón de diseño MVVM - Leomaris Reyes – Medium. *Medium.* https://medium.com/@reyes.leomaris/aplicando-el-patr%C3%B3n-de-dise%C3%B1o-mvvm-d4156e51bbe5

[15] Diego.Coder. (2023, 2 septembre). Introducción a las "Clean Architectures" - diego.coder - Medium. *Medium.* https://medium.com/@diego.coder/introducci%C3%B3n-a-las-clean-architectures-723fe9fe17fa

[16] *Inserción de dependencias con Hilt. (s. f.). Android Developers. https://developer.android.com/training/dependency-injection/hilt-android?hl=es-419*

[17] JavaScript | MDN. (2023, 24 julio). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/JavaScript

[18] Firebase Authentication. (s. f.). Firebase. https://firebase.google.com/docs/auth?hl=es

[19] Almacenamiento en la nube para Firebase | Cloud Storage for Firebase. (s. f.). Firebase. https://firebase.google.com/docs/storage?hl=es

[20] JSON. (s. f.). https://www.json.org/json-es.htm

[21] Firestore | Firebase. (s. f.). Firebase. https://firebase.google.com/docs/firestore?hl=es

[22] Firebase Crashlytics. (s. f.). Firebase. https://firebase.google.com/docs/crashlytics?hl=es

APÈNDIX

A1. PLANIFICACIÓ

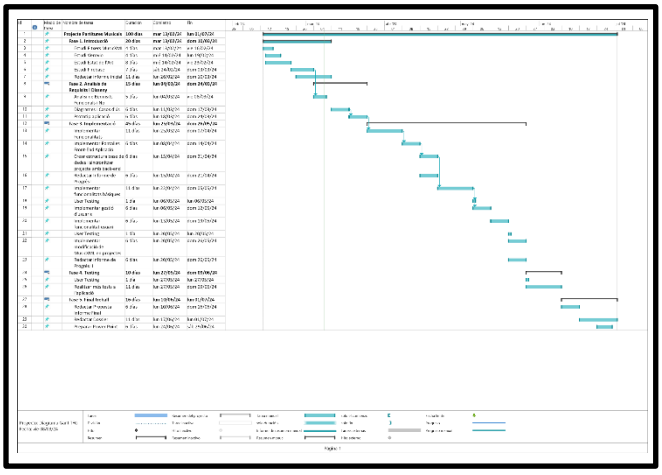

Fig. 16. Diagrama de Gantt del projecte

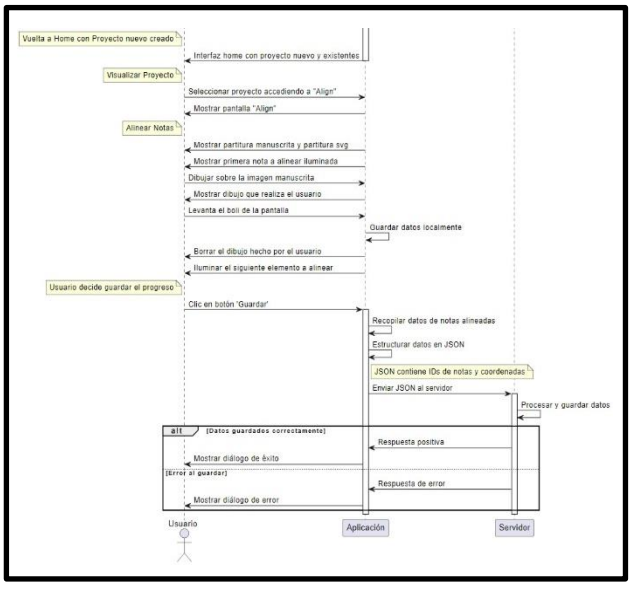

Fig. 14. Diagrama de seqüència, part 2

A2. DISSENY

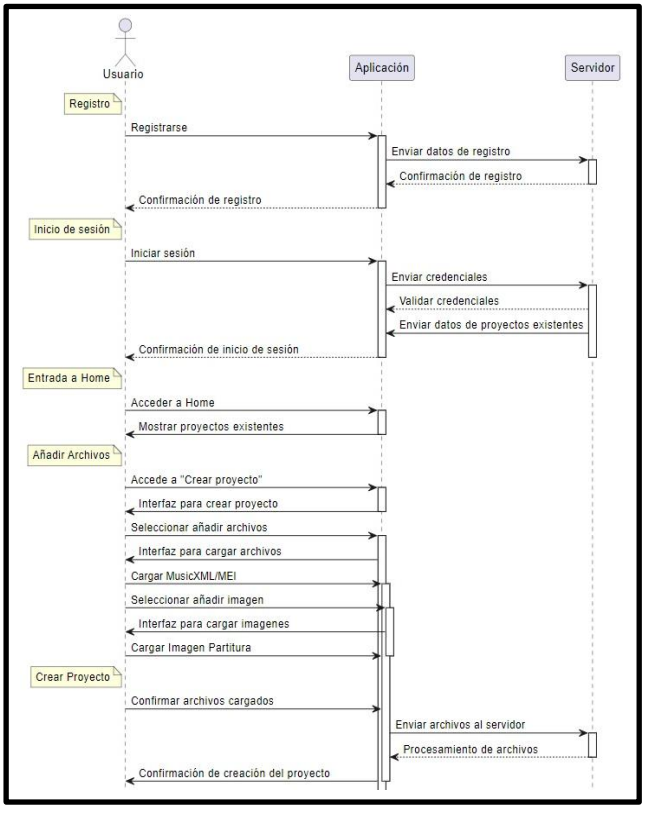

Fig. 13. Diagrama de seqüència, part 1

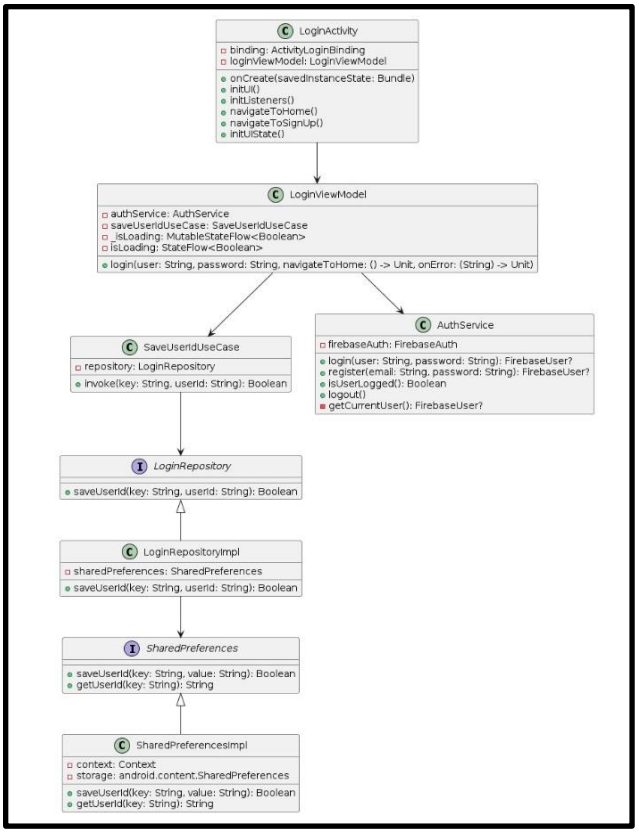

Fig. 15. Diagrama de classes de la pantalla de *login*

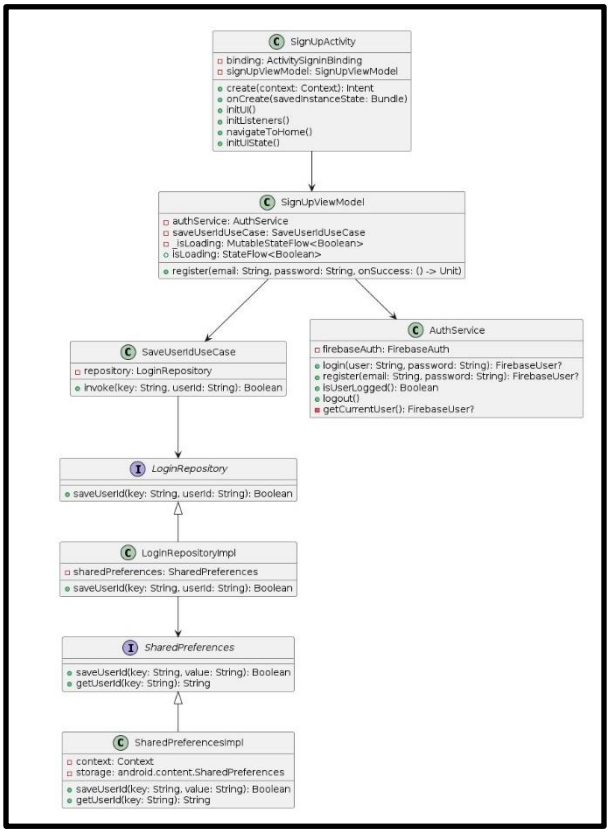

Fig. 16. Diagrama de classes de la pantalla de registre

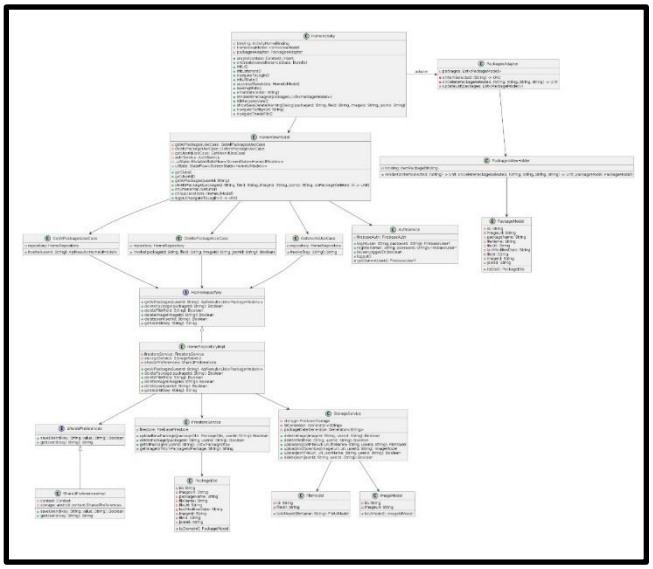

Fig. 17. Diagrama de classes de la pantalla *home*

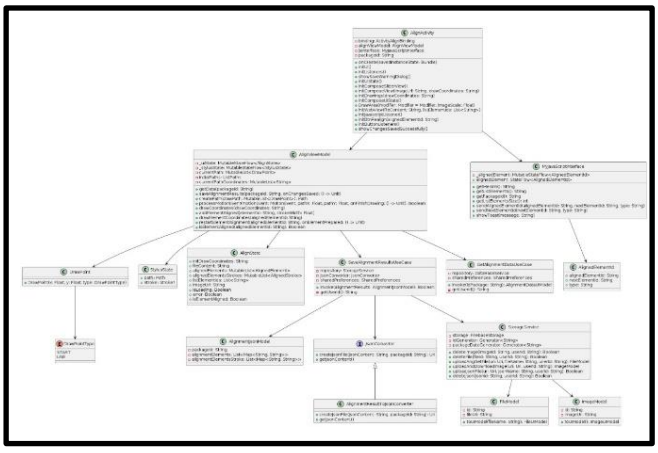

Fig. 18. Diagrama de classes de la pantalla d'alineament

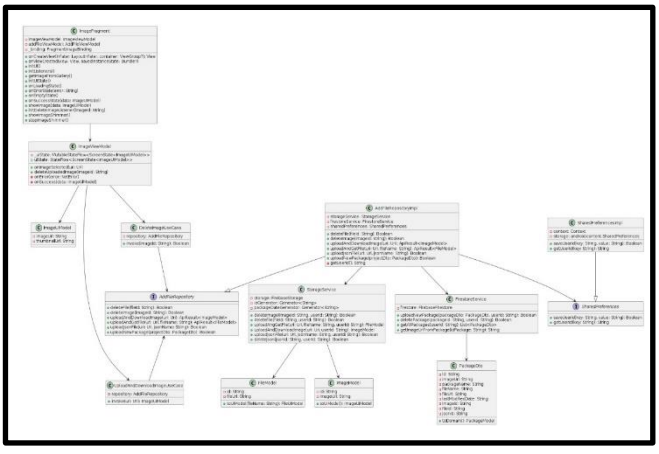

Fig. 19. Diagrama de classes de la selecció d'imatges a la pantalla de creació de projectes

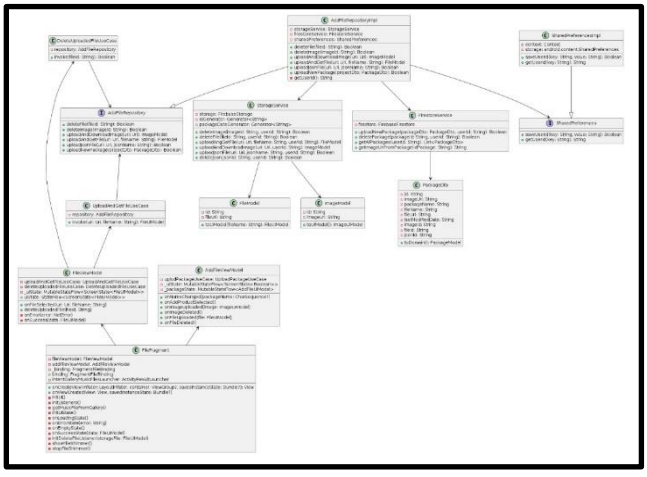

Fig. 20. Diagrama de classes de la selecció de fitxers a la pantalla de creació de projectes

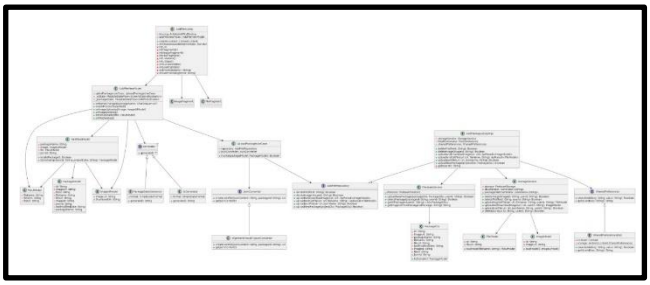

Fig. 21. Diagrama de classes de la pantalla de creació de projectes

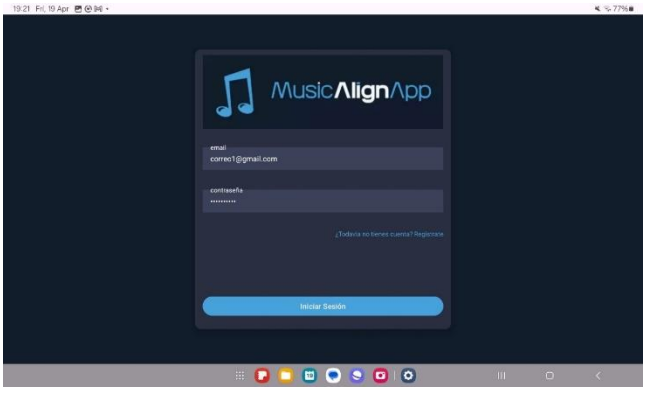

Fig. 22. Primera versió interfície de la pantalla *login*

Fig. 23. Primera versió interfície de la pantalla de registre

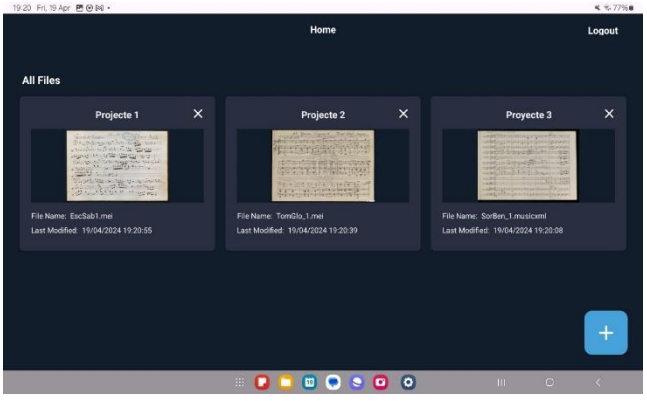

Fig. 24. Primera versió interfície de la pantalla *home*

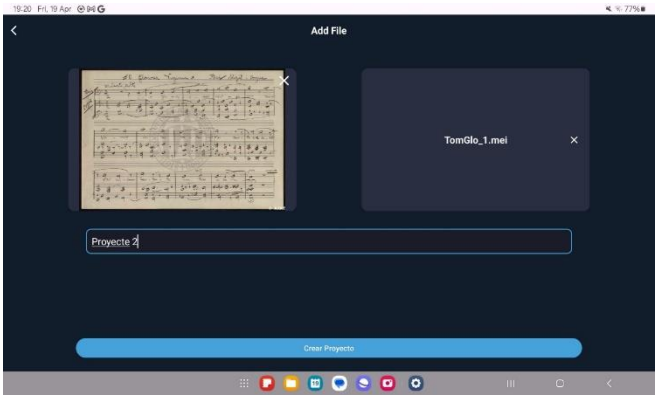

Fig. 25. Primera versió interfície de la pantalla de creació de projectes

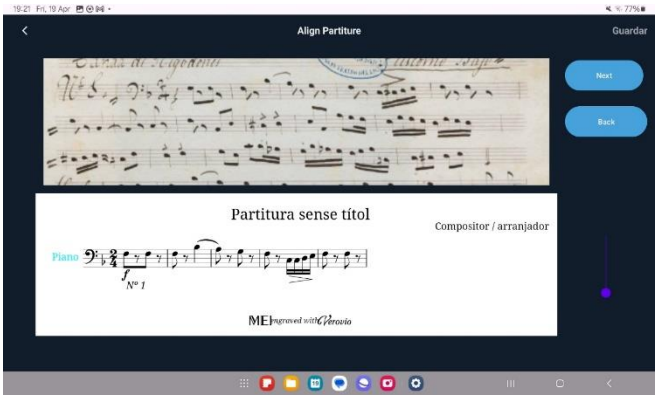

Fig. 26. Primera versió interfície de la pantalla d'alineament