
This is the **published version** of the bachelor thesis:

Escoda Matas, Judit. *Creació d'una aplicació per validar partitures musicals*.
Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026
(Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334391>

under the terms of the  license.

Creació d'una aplicació per validar partitures musicals

Judit Escoda Matas

Resum — El Reconeixement Òptic Musical (OMR) és el procés de transcriure partitures musicals digitalitzades a formats editables com MusicXML o MIDI, però les eines actuals generen errors que requereixen una revisió manual. Aquest treball presenta el disseny i desenvolupament d'una aplicació d'escriptori per facilitar aquesta tasca de validació. L'eina rep com a entrades una imatge de la partitura original i el fitxer MusicXML generat per un sistema OMR, i ofereix a l'usuari una interfície on es carreguen simultàniament la imatge original i la partitura digital generada, permetent comparar-les i corregir els errors detectats. La interfície integra l'editor MuseScore 4 com a component incrustat, de manera que les correccions es poden fer de forma nativa sense abandonar l'aplicació. La persistència de les sessions de validació es gestiona amb una base de dades SQLite, permetent reprendre el treball interromput més endavant.

L'aplicació s'ha implementat en Python amb PyQt6 i segueix una arquitectura modular que separa la interfície gràfica, el back end de processament i la capa de persistència. El resultat és una eina funcional que cobreix el flux complet de validació: des de la càrrega de la sortida de l'OMR fins a l'exportació del MusicXML corregit.

Paraules clau — Reconeixement Òptic Musical (OMR), validació, partitures musicals, MusicXML, MuseScore4, PyQt6.

Resumen — El Reconocimiento Óptico Musical (OMR) es el proceso de transcribir partituras musicales digitalizadas a formatos editables como MusicXML o MIDI, pero las herramientas actuales generan errores que requieren una revisión manual. Este trabajo presenta el diseño y desarrollo de una aplicación de escritorio para facilitar esta tarea de validación. La herramienta recibe como entradas una imagen de la partitura original y el archivo MusicXML generado por un sistema OMR, y ofrece al usuario una interfaz donde se cargan simultáneamente la imagen original y la partitura digital generada, permitiendo compararlas y corregir los errores detectados. La interfaz integra el editor MuseScore 4 como componente incrustado, de manera que las correcciones se pueden hacer de forma nativa sin abandonar la aplicación. La persistencia de las sesiones de validación se gestiona con una base de datos SQLite, permitiendo retomar el trabajo interrumpido más adelante.

La aplicación se ha implementado en Python con PyQt6 y sigue una arquitectura modular que separa la interfaz gráfica, el backend de procesamiento y la capa de persistencia. El resultado es una herramienta funcional que cubre el flujo completo de validación: desde la carga de la salida del OMR hasta la exportación del MusicXML corregido.

Palabras clave — Reconocimiento Óptico Musical (OMR), validación, partituras musicales, MusicXML, MuseScore4, PyQt6.

Abstract — Optical Music Recognition (OMR) is the process of transcribing digitized musical scores into editable formats such as MusicXML or MIDI, but current tools generate errors that require manual revision. This work presents the design and development of a desktop application to facilitate this validation task. The tool takes as input an image of the original score and the MusicXML file generated by an OMR system, and provides the user with an interface where both the original image and the generated digital score are loaded simultaneously, allowing them to be compared and the detected errors to be corrected. The interface integrates the MuseScore 4 editor as an embedded component, so that corrections can be made natively without leaving the application. The persistence of validation sessions is managed with a SQLite database, allowing interrupted work to be resumed at a later time.

The application has been implemented in Python with PyQt6 and follows a modular architecture that separates the graphical interface, the processing backend, and the persistence layer. The result is a functional tool that covers the complete validation workflow: from loading the OMR output to exporting the corrected MusicXML.

Index Terms—Optical Music Recognition (OMR), validation, musical scores, MusicXML, MuseScore4, PyQt6.

-
- E-mail de contacte: Judith.EscodaM@autonoma.cat
 - Menció realitzada: Enginyeria del Software
 - Treball tutoritzat per: Alicia Fornes Bisquerra
 - Curs 2025/26

1 INTRODUCCIÓ - CONTEXT DEL TREBALL

El Reconeixement Òptic Musical (OMR) és el camp que té com a objectiu reconèixer i interpretar imatges de partitures musicals i convertir-les en formats editables, com MIDI o MusicXML [1]. Aquesta digitalització permet reproduir, analitzar i editar grans col·leccions de música escrita.

Actualment, existeixen eines comercials i de codi obert, com Audiveris [2], que permeten digitalitzar partitures amb un nivell de precisió elevat. No obstant això, aquestes solucions presenten limitacions importants quan es tracta de partitures manuscrites o d'imatges capturades amb mòbils, on la variabilitat de l'escriptura o la qualitat incrementen els errors.

Encara que les tècniques basades en aprenentatge han millorat la fase de reconeixement, els errors segueixen sent habituals i, en conseqüència, la validació i correcció manual dels resultats generats és una etapa necessària en molts dels casos.

En aquest context, aquest Treball de Final de Grau se situa com una peça intermèdia entre l'OMR i l'usuari final, proposant una aplicació centrada en la validació i correcció dels resultats de reconeixement.

2 OBJECTIUS

L'objectiu principal ha estat dissenyar i desenvolupar una aplicació que permeti visualitzar, validar i corregir de manera interactiva els resultats generats per un sistema de Reconeixement Òptic Musical, prenent com a entrada fitxers MusicXML generats per una eina OMR existent i la partitura.

A partir d'aquest objectiu principal, es van definir els objectius específics següents:

- Llegir i processar la sortida d'un sistema OMR i mostrar-ne el contingut d'una forma comprensible per l'usuari final.
- Permetre la correcció interactiva d'errors de reconeixement (notes, durades, silencis, clau, armadura, etc.).
- Permetre exportar el resultat validat a formats estàndards com MusicXML.
- Documentar el procés de desenvolupament seguint les activitats fonamentals de l'enginyeria del software.
- Avaluar l'aplicació mitjançant proves amb usuaris del domini musical.

En conjunt, tots aquests objectius han guiat totes les decisions de disseny i implementació preses al llarg del projecte, i tots han estat assolits en la seva totalitat.

3 ESTAT DE L'ART

3.1 El Reconeixement Òptic Musical

L'OMR és un camp d'investigació que té com a objectiu llegir computacionalment la notació musical en documents [3]. Tot i els avenços, encara no existeix una definició unificada del problema ni una avaluació estandarditzada que permeti comparar els diferents enfocaments de manera senzilla.

Els sistemes actuals funcionen bé amb partitures impreses, però els errors augmenten significativament amb partitures manuscrites o amb una notació més complexa (com múltiples pentagrames i veus). Aquesta limitació és precisament el que fa necessària una fase de validació manual posterior al reconeixement, que és el que aborda aquest treball.

3.2 Eines i tecnologies analitzades

Durant la fase de disseny es van analitzar diverses eines i tecnologies relacionades amb la manipulació i representació de música digital:

MusicXML [1]: format obert i estàndard per la representació de partitures musicals. Permet emmagatzemar de manera estructurada notes, compassos, claus, armadures i altres elements de notació. S'ha utilitzat com a format d'entrada i de sortida de l'aplicació.

Humdrum [4]: entorn de programari per l'anàlisi computacional de música. Inclou el format kern, una representació textual estructurada de partitures. Es va analitzar però es va descartar per la seva menor adopció en eines OMR actuals.

KernPy [5]: llibreria Python que facilita la manipulació de fitxers en format kern de Humdrum. Igual que Humdrum, es va descartar.

music21 [6]: llibreria Python per a l'anàlisi i manipulació de música en formats digitals. Permet llegir, escriure i processar estructures musicals en MusicXML. S'ha utilitzat com a nucli del backend per generar i exportar els fitxers MusicXML de l'aplicació.

Verovio [7]: llibreria de renderització de partitures de MusicXML a formats gràfics. Es va considerar com a opció per mostrar la partitura dins de l'aplicació, però es va descartar perquè no ofereix funcionalitats d'edició interactiva, que és un requisit clau de l'eina.

MuseScore [8]: editor de partitures de codi obert amb interfície gràfica completa. S'ha escollit com a component de visualització i edició perquè ofereix totes les funcionalitats

d'edició musical de manera nativa, evitant haver de reimplementar-les des de zero.

PyQt6 [9]: llibreria Python per al desenvolupament d'interfícies gràfiques d'escriptori basada en Qt. S'ha escollit per la seva robustesa i perquè permet accedir al sistema de finestres natiu del sistema operatiu, cosa necessària per a la integració de MuseScore. Altres alternatives es van descartar per les seves limitacions en la gestió de gràfics i finestres.

SQLite [10]: base de dades lleugera i sense servidor que s'integra directament dins de l'aplicació. S'ha escollit per la persistència de les sessions de validació per la seva simplicitat i per no requerir cap configuració addicional per part de l'usuari.

De les eines analitzades, s'han descartat Humdrum i KernPy per la seva menor adopció en el context de l'OMR i per la complexitat afegida que suposarien sense un benefici clar. Verovio es va descartar per la manca de funcionalitats d'edició interactiva. Les eines seleccionades finalment (MusicXML, music21, MuseScore, PyQt6 i SQLite) formen un stack cohesionat que cobreix totes les necessitats de l'aplicació: representació estàndard de partitures, manipulació programàtica, edició nativa, interfície gràfica robusta i persistència lleugera.

4 METODOLOGIA I PLANIFICACIÓ

El projecte s'ha desenvolupat seguint una metodologia incremental i iterativa, construint versions successives de l'aplicació a partir d'un prototip mínim viable.

El desenvolupament s'ha organitzat en cinc fases descrites a continuació.

Anàlisi de requisits

En primer lloc es van definir els requisits funcionals i no funcionals de l'aplicació a partir de les reunions realitzades sobre el tema. Els requisits funcionals cobreixen des de la càrrega de la imatge i la importació del fitxer MusicXML fins a la correcció interactiva d'errors, la gestió de sessions i l'exportació del resultat. També es van definir requisits no funcionals relacionats amb la usabilitat i el rendiment de l'aplicació. A partir d'aquests requisits es van elaborar els casos d'ús principals.

Disseny

Un cop definits els requisits, es va dissenyar l'arquitectura del sistema dividida en tres capes: la interfície gràfica, el backend de processament i la capa de persistència. Es va elaborar un mockup de la pantalla principal amb Figma, que va servir per validar la disposició dels tres panells abans de començar la implementació. També es van elaborar diagrames de casos d'ús i de seqüència per descriure les interaccions entre l'usuari, l'aplicació, MuseScore i la base de dades. Finalment, es van prendre les decisions tecnològiques definitives descrites anteriorment.

Implementació

El desenvolupament es va fer de manera incremental, començant per una versió bàsica que permetia carregar una imatge i un fitxer MusicXML i visualitzar-los simultàniament, i afegint progressivament les funcionalitats de validació, correcció i exportació. Tot el codi s'ha gestionat amb Git i GitHub, amb commits regulars que reflecteixen l'evolució del projecte.

Proves

Es van implementar proves unitàries i d'integració dels components principals: el parser de MusicXML i el motor de validació. A més, es van realitzar proves funcionals amb partitures reals per verificar el comportament de l'aplicació en condicions d'ús real. Actualment, les proves amb usuaris finals estan pendents de dur-se a terme per poder avaluar la usabilitat de l'eina.

Documentació

Al llarg de tot el projecte s'han elaborat dos informes de seguiment que han permès consignar l'estat del desenvolupament, els ajustos efectuats respecte a la planificació inicial i les decisions tècniques més rellevants. La memòria final recull tot el procés de manera completa.

Planificació

El projecte s'ha organitzat en cinc fases distribuïdes al llarg del curs acadèmic. La primera fase, de tres setmanes, va estar dedicada a la revisió d'eines i la definició de requisits. La segona, també de tres setmanes, al disseny de l'arquitectura i la interfície. La tercera, de cinc setmanes, a la implementació funcional bàsica. La quarta, de cinc setmanes, a les funcionalitats de validació, correcció i exportació. I la cinquena i última, de quatre setmanes, a les proves amb usuaris i la redacció de la memòria.

En general, s'ha seguit la planificació prevista. L'únic ajust rellevant ha estat en la fase d'implementació, on la integració de MuseScore va requerir més temps del previst inicialment a causa de l'absència d'una API oficial, cosa que va obligar a buscar i implementar una solució alternativa.

A la figura 1 es mostra el diagrama de Gantt amb la distribució de les fases del projecte.

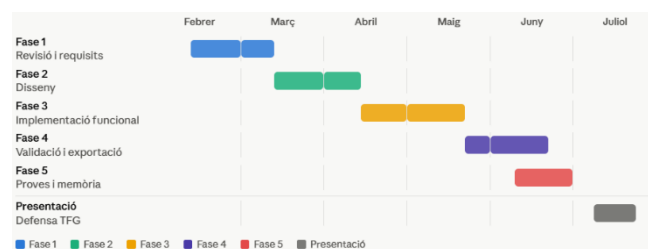


Figura 1: Diagrama de Gantt

5 REQUISITS

Durant la fase inicial del projecte es van definir els requisits funcionals i no funcionals amb l'objectiu de guiar el

disseny i el desenvolupament de l'aplicació, tenint en compte les necessitats de l'usuari final i els objectius generals del treball. Es van identificar requisits funcionals que cobreixen les funcionalitats principals de l'eina: la càrrega de la imatge i del fitxer MusicXML, la visualització simultània de les dues representacions, la gestió i navegació pels errors detectats, la correcció interactiva d'elements musicals, la persistència de sessions i l'exportació del resultat validat. A més, es van definir requisits no funcionals relacionats amb la usabilitat i el rendiment de l'aplicació. Els requisits es van mantenir estables al llarg de tot el desenvolupament, sense canvis significatius respecte a la definició inicial.

A les següents taules es poden consultar els requisits funcionals i no funcionals.

Taula de requisits funcionals:

ID	Descripció
RF-01	L'aplicació ha de permetre carregar una imatge de la partitura original (escanejada o fotografia).
RF-02	L'aplicació ha de permetre importar fitxers en format <i>MusicXML</i> generats per un sistema de reconeixement OMR.
RF-03	L'aplicació ha de mostrar simultàniament la imatge original i la representació digital rendritzada de la partitura amb una disposició que permeti la comparació directa.
RF-04	L'aplicació ha de permetre mostrar i gestionar els errors i elements amb baixa confiança proporcionats pel sistema OMR.
RF-05	L'usuari ha de poder seleccionar un error/dubte i veure la seva localització exacta al <i>MusicXML</i> .
RF-06	L'aplicació ha de permetre corregir elements musicals (notes, durades, silencis, claus, armadures, etc.).
RF-07	L'aplicació ha de permetre navegar entre errors i consultar l'estat global de validació.
RF-08	L'aplicació ha de permetre desar i reprendre una sessió de validació, conservant les correccions i marques.
RF-09	L'aplicació ha de permetre fer zoom i desplaçament tant per sobre de la imatge original com per sobre de la partitura digital.
RF-10	L'aplicació ha de permetre exportar el resultat validat en format <i>MusicXML</i> .

RF-11	L'aplicació ha de proporcionar un resum bàsic d'estadístiques de validació (per exemple, nombre dels errors corregits/confirmats).
-------	--

Taula de requisits no funcionals:

ID	Descripció
RNF-01	La interfície de l'aplicació ha de permetre identificar els errors ràpidament i corregir-los amb el mínim nombre de passos possibles.
RNF-02	Els errors s'han de mostrar amb codis visuals consistents i han de ser fàcilment diferenciables.
RNF-03	L'aplicació ha de respondre de manera fluida en operacions de navegació, selecció i visualització d'errors.
RNF-04	L'aplicació ha de ser usable i intuïtiva, permetent que l'usuari pugui realitzar una validació simple sense formació prèvia.

6 DIAGRAMES D'ÚS

Els casos d'ús descriuen les interaccions principals entre els actors del sistema i l'aplicació. S'han identificat tres actors: l'usuari, que és la persona que realitza la validació de la partitura; MuseScore, que actua com a component extern de visualització i edició; i la base de dades, que gestiona la persistència de la informació.

Els casos d'ús principals són els següents:

- Carregar la imatge de la partitura original.
- Importar el fitxer MusicXML generat pel sistema OMR.
- Visualitzar simultàniament la imatge i la partitura digital.
- Navegar pels errors detectats.
- Localitzar visualment un error a la partitura.
- Corregir un element musical.
- Desar una sessió de validació.
- Reprendre una sessió prèvia.
- Exportar la partitura validada.
- Consultar el resum d'estadístiques de validació.

Seguidament trobareu la taula amb tots els casos d'ús:

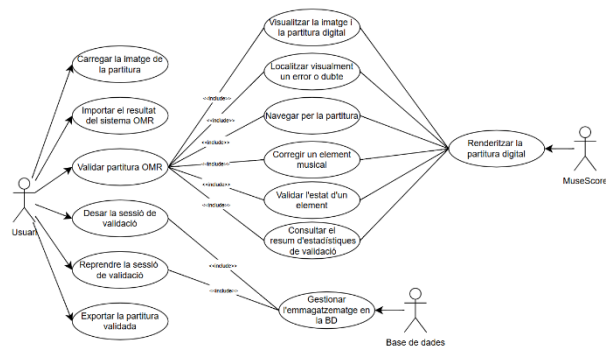
ID	Cas d'ús	Actor	Descripció
UC-01	Carregar la imatge de la partitura	Usuari	L'usuari selecciona una imatge escanejada o fotografia de la partitura perquè

			l'aplicació la mostri a la zona de visualització.
UC-02	Importar el resultat del sistema OMR (MusicXML)	Usuari	L'usuari importa un fitxer MusicXML generat per un sistema OMR. L'aplicació carrega el contingut musical i la informació d'errors/baixa confiança associada.
UC-03	Visualitzar la imatge i la partitura digital	Usuari	L'aplicació mostra simultàniament la imatge original i la representació digital renderitzada de la partitura, amb una disposició que permeti la comparació directa.
UC-04	Navegar per la partitura	Usuari	L'usuari pot desplaçar-se per la partitura i recórrer els errors, veient l'estat global de la validació.
UC-05	Localitzar visualment un error o dubte	Usuari	En seleccionar un error o element amb baixa confiança, l'aplicació resalta la seva posició a la partitura digital, permetent identificar la seva posició exacta.
UC-06	Corregir un element musical	Usuari	L'usuari modifica un element musical proposat per l'OMR, actualitzant el model intern i la visualització.
UC-07	Desar sessió de validació	Usuari	L'usuari desa la sessió de validació actual a la base de dades, per poder continuar més endavant.
UC-08	Reprendre sessió de validació	Usuari	L'usuari consulta sessions prèvies i en repren una, recuperant la imatge, el fitxer MusicXML i l'estat de validació per poder continuar el treball.
UC-09	Exportar la	Usuari	Un cop finalitzada la

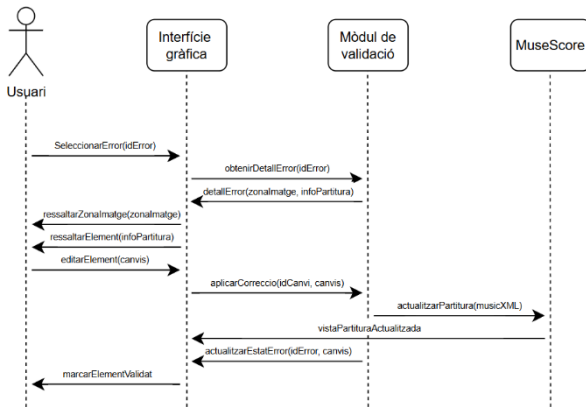
	partitura validada		validació, l'usuari exporta el resultat a un fitxer MusicXML actualitzat amb les correccions.
UC-10	Consultar el resum d'estadístiques de validació	Usuari	L'usuari consulta un resum bàsic d'estadístiques.
UC-11	Gestionar emmagatzematge en la BD	BD	L'aplicació guarda i recupera informació de sessions, partitures i estadístiques en una base de dades.
UC-12	Renderitzar la partitura amb MuseScore	MuseScore	L'aplicació envia el MusicXML a MuseScore i obté una vista actualitzada de la partitura.

Seguidament trobareu el diagrama de casos d'ús i el diagrama de seqüència respectivament. El diagrama de seqüència descriu en detall el flux d'interaccions entre l'usuari, la interfície gràfica, el mòdul de validació i MuseScore durant el procés de localització i correcció d'un error OMR.

6.1 Diagrama de casos d'ús



6.2 Diagrama de seqüència



original detectat, el valor corregit per l'usuari, la confiança de l'OMR, l'estat de validació i les coordenades de la seva posició a la imatge original.

- La taula canvis registra l'historial complet de correccions. Cada vegada que l'usuari modifica un element, es crea un registre amb el camp modificat, el valor anterior i el valor nou, cosa que permet traçar tota l'evolució de la validació.

7.3 Disseny de la interfície

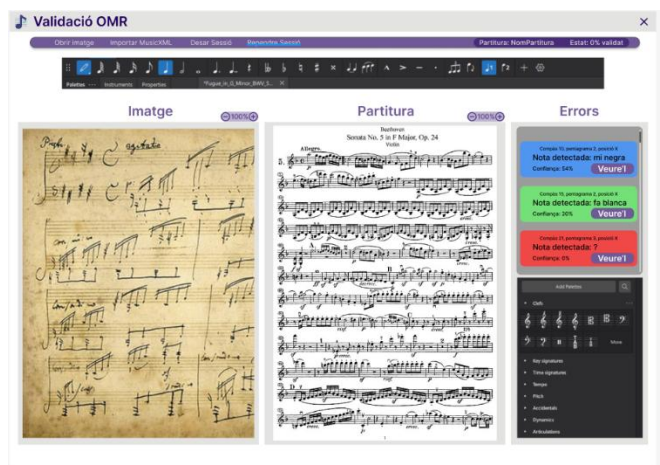
La interfície gràfica s'ha dissenyat seguint un criteri de claredat i usabilitat, amb l'objectiu que l'usuari pugui identificar i corregir errors de manera ràpida i sense formació prèvia.

La finestra principal es divideix en tres panells, on es pot modificar la seva disposició segons les preferències de l'usuari.

Aquesta disposició permet comparar directament la imatge original amb la partitura digital i consultar els errors detectats de manera simultània, sense haver de canviar de vista.

- El panell esquerre mostra la imatge original de la partitura.
- El panell central mostra la partitura digital integrant MuseScore 4 directament dins de l'aplicació.
- El panell dret mostra la llista d'errors detectats en forma de targetes.

El disseny inicial de la interfície es va prototipar amb Figma abans de començar la implementació.



Imatge 1: Mockup inicial

7 DISSENY

7.1 Arquitectura del sistema

L'aplicació segueix una arquitectura de tres capes clarament separades, de manera que cada capa té una responsabilitat ben definida i les modificacions en una no afecten les altres.

La primera capa és la interfície gràfica, implementada amb PyQt6, que s'encarrega de mostrar la informació a l'usuari i gestionar les seves interaccions. La segona és el backend de processament, que conté tota la lògica de l'aplicació: la lectura de la sortida de l'OMR, la conversió a MusicXML, el motor de validació i el motor de correccions. La tercera és la capa de persistència, que gestiona l'emmagatzematge i la recuperació de la informació mitjançant una base de dades SQLite local.

El flux principal de l'aplicació és el següent: l'usuari carrega la imatge original i la sortida del sistema OMR; el backend processa aquesta sortida i genera una representació interna en format MusicXML; el motor de validació identifica els elements amb baixa confiança i els marca com a pendents de revisió; l'usuari els revisa i corregeix des de la interfície; i finalment s'exporta el MusicXML validat amb les correccions incorporades.

7.2 Disseny de la base de dades

La persistència de la informació s'implementa amb una base de dades SQLite local estructurada en tres taules relacionades entre elles.

- La taula sessions emmagatzema les metadades de cada sessió de validació: el nom, la data de creació i modificació, l'estat global i les rutes als fitxers de la imatge i el MusicXML associats.
- La taula elements emmagatzema cada element detectat pel sistema OMR associat a una sessió. Per a cada element es guarda el tipus, la ubicació a la partitura (compàs, pentagrama i posició), el valor

8 IMPLEMENTACIÓ

8.1 Tecnologies utilitzades

L'aplicació s'ha implementat en Python, aprofitant la seva versatilitat i la disponibilitat de llibreries especialitzades tant per al processament de música com per al

desenvolupament d'interfícies gràfiques.

Les tecnologies principals utilitzades són: PyQt6 per a la interfície gràfica d'escriptori, music21 per a la generació i manipulació de fitxers MusicXML, SQLite com a base de dades local per a la persistència de sessions, i MuseScore 4 com a component extern de visualització i edició de partitures. A més, s'han utilitzat les API Win32 de Windows per a la integració de MuseScore dins de la interfície, mitjançant la llibreria pywin32.

8.2 Estructura del projecte

El projecte s'organitza en tres carpetes principals que reflecteixen les tres capes de l'arquitectura. La carpeta *ui* conté els mòduls de la interfície gràfica. La carpeta *backend* conté tota la lògica de processament. La carpeta *db* conté el gestor de la base de dades i l'esquema SQL.

A l'arrel del projecte hi ha el fitxer *main.py*, que inicialitza la base de dades i llança la finestra principal, i el fitxer *config.py*, que centralitza les configuracions globals com la ruta a MuseScore i els directoris de treball.

8.3 Backend

El backend està format per quatre mòduls principals. El *MusicXMLParser* usa music21 per carregar el fitxer i extreure tots els elements musicals amb la seva ubicació (compàs, pentagrama i posició ordinal dins el compàs). El *ValidationEngine* processa els elements i els assigna una confiança OMR simulada, marcant com a pendents de revisió aquells que superen el llindar establert per l'usuari. El *SessionManager* gestiona la creació i recuperació de sessions a la base de dades. El *CorrectionEngine* aplica les correccions de l'usuari i les registra a l'historial de canvis.

8.4 Interfície gràfica

La interfície s'ha implementat amb PyQt6 i es divideix en tres panells. El panell d'imatge mostra la partitura original amb suport de zoom i desplaçament. El panell de partitura mostra la representació digital generada amb MuseScore incrustat. El panell d'errors presenta els elements pendents de revisió en forma de targetes. Un slider a la part superior permet filtrar els elements per confiança OMR en temps real, de manera que l'usuari pot centrar-se primer en els elements on el sistema OMR era menys segur. L'aplicació ofereix també dos layouts intercanviables: tres columnes horitzontals o imatge i partitura apilades verticalment amb els errors a la dreta.

8.5 Integració de MuseScore

Actualment MuseScore no ofereix cap API pública per comunicar-s'hi des d'una aplicació externa. La integració s'ha resolt en dues parts.

La primera és l'incrustació visual: l'aplicació llança

MuseScore com un procés independent, localitza la seva finestra mitjançant les API Win32 de Windows i la incrusta dins del panell central. El resultat és que l'usuari veu una sola finestra però hi ha dues aplicacions funcionant alhora.

La segona és la navegació programàtica: quan l'usuari prem el botó "Veure'l" en una targeta d'error, l'aplicació ha de moure el cursor de selecció de MuseScore fins a la nota concreta. Com que MuseScore tampoc ofereix cap API per a això, s'ha implementat mitjançant PostMessage Win32, que envia missatges de teclat directament a la cua de missatges del procés de MuseScore sense necessitar que tingui el focus.

9 TESTING

Durant tot el desenvolupament s'han realitzat proves contínues per verificar el correcte funcionament de les funcionalitats implementades i detectar errors de manera progressiva. S'han implementat proves unitàries dels components principals del backend, cobrint el parsejament de MusicXML, el motor de validació, el gestor de la base de dades i el gestor de sessions.

A nivell funcional, s'ha provat l'aplicació amb tres partitures reals amb els seus fitxers MusicXML.

Les operacions principals de l'aplicació responen de manera fluida. L'única excepció és la incrustació de MuseScore, que requereix uns segons de càrrega inicial, però es gestiona de forma asíncrona per no bloquejar la interfície. Es va dur a terme una sessió de prova amb un usuari del domini musical per avaluar la usabilitat de l'eina, els resultats els quals es detallen a la següent secció.

10 FEEDBACK DE L'USUARI

Per avaluar la usabilitat de l'aplicació en un context real, es va dur a terme una sessió de prova amb un estudiant de Musicologia de quart curs, l'Edgar Zamora.

Aspectes valorats positivament

L'usuari va destacar que la interfície és intuïtiva: sense cap explicació prèvia, va entendre que havia de carregar la imatge i el MusicXML i que els errors apareixien al panell de la dreta. El botó "Veure'l" va ser especialment ben valorat, ja que MuseScore navega directament al compàs on hi ha l'error sense que l'usuari l'hagi de buscar manualment, cosa que estalvia molt de temps en partitures llargues.

També es va valorar positivament la barra de progrés a l'apartat 'Estadístiques', ja que dona sensació de control sobre la quantitat de treball que queda pendent, i la possibilitat de reprendre la sessió més endavant amb tots els canvis guardats.

Dificultats identificades

L'usuari va trobar que inicialment no tenia clar quin fitxer

havia de carregar primer. El filtre de confiança tampoc era evident al principi: no resultava obvi quin criteri fer servir per decidir el llindar. Finalment, va notar que la imatge original i MuseScore no sempre estaven sincronitzats visualment, de manera que de vegades costava localitzar l'error a la imatge escanejada.

Suggeriments recollits

L'usuari va proposar tres millores: la possibilitat d'afegir una nota de text lliure a cada error per distingir tipus d'incidències, botons de navegació seqüencial ("Anterior / Següent error") per evitar haver de fer scroll al panell, i una vista resum final abans d'exportar que mostri tots els canvis efectuats. Aquests suggeriments s'han incorporat a les línies de treball futur.

Valoració global

L'usuari va valorar l'eina molt positivament, destacant que redueix significativament el temps de revisió respecte a fer-la manualment compàs per compàs a MuseScore.

11 RESULTATS

11.1 Inici de sessió

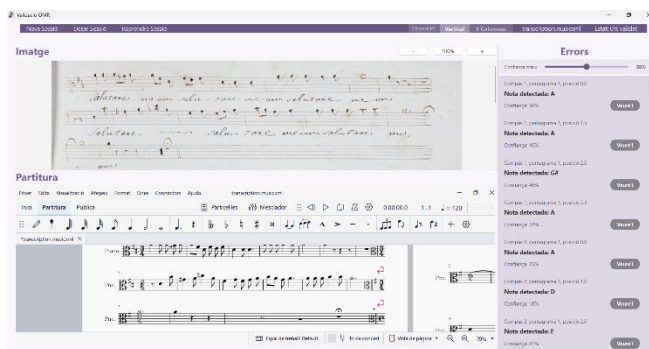
Per iniciar una sessió de validació, l'usuari selecciona la imatge original de la partitura i el fitxer MusicXML generat pel sistema OMR. L'aplicació parseja el MusicXML amb music21, extreu tots els elements musicals amb la seva ubicació i els desa a la base de dades associats a la sessió nova. A partir d'aquí es carreguen els tres panells simultàniament i l'usuari pot començar la revisió. El nom de la sessió s'assigna automàticament a partir del nom del fitxer MusicXML i es pot editar en qualsevol moment.

11.2 Visualització simultània

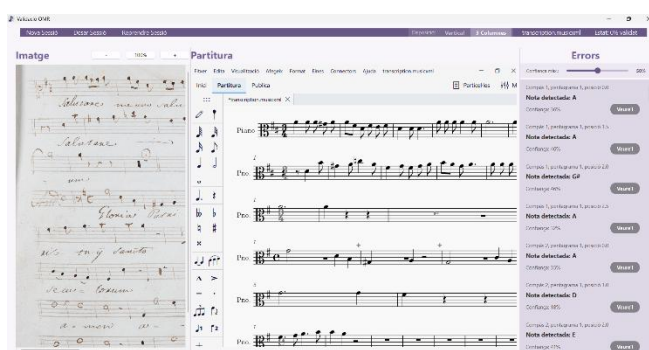
La finestra principal mostra simultàniament la imatge original de la partitura, la representació digital a MuseScore i la llista d'errors detectats. Aquesta disposició permet comparar directament la imatge amb la partitura digital sense haver de canviar de finestra, que és precisament l'objectiu principal de l'eina.

L'aplicació ofereix dues disposicions intercanviables sense perdre cap dada: tres columnes horitzontals, on els tres panells es mostren un al costat de l'altre, i disposició vertical, on la imatge i la partitura s'apilen verticalment a l'esquerra i els errors queden a la dreta. El canvi entre disposicions és immediat.

El panell d'imatge suporta zoom amb la roda del ratolí i desplaçament per arrossegament, amb botons addicionals per ampliar, reduir i restablir el zoom al 100%.



Imatge 2: Panells en disposició vertical



Imatge 3: Panells en disposició horitzontal

11.3 Filtratge per confiança

El panell d'errors inclou un slider que permet filtrar els elements per confiança OMR. En desplaçar el slider, la llista s'actualitza en temps real mostrant únicament els elements amb una confiança igual o inferior al llindar seleccionat. Amb un valor per defecte del 50%, l'usuari veu només els elements on el sistema OMR era menys segur, reduint significativament el volum d'elements a revisar sense perdre els errors més probables.

Cada element es mostra com una targeta que inclou la seva ubicació a la partitura (compàs, pentagrama i posició), el valor detectat per l'OMR i el percentatge de confiança.



Imatge 4: Panell "Errors"

11.4 Navegació a MuseScore

En prémer el botó "Veure'l" d'una targeta, l'aplicació navega automàticament fins a la nota corresponent dins de MuseScore i mostra un badge a la capçalera del panell de partitura amb la posició exacta de l'element (compàs, pentagrama i posició).

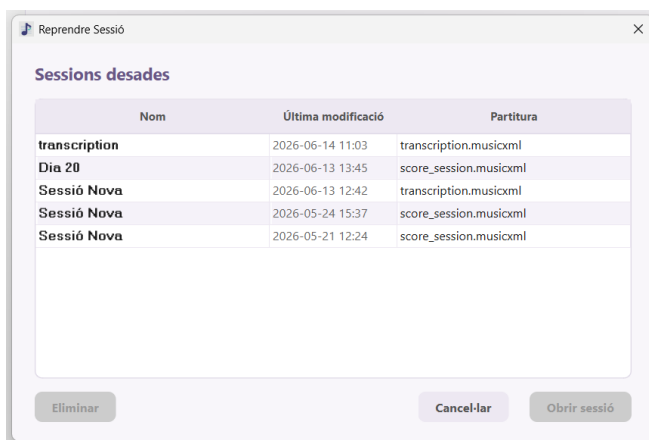


Imatge 5: Navegació automàtica MuseScore

11.5 Gestió de sessions

L'aplicació permet desar l'estat de la validació en qualsevol moment i reprendre-la més endavant. En desar una sessió, l'usuari pot editar-ne el nom. En reprendre-la, el sistema comprova que els fitxers originals existeixin al disc i restaura tots els panells amb l'estat previ de cada element.

El diàleg de sessions mostra totes les sessions desades ordenades per data de modificació, amb el nom, la data i el fitxer MusicXML associat. Des d'aquest diàleg també es poden eliminar sessions, cosa que esborra tots els elements i canvis associats.



Imatge 6: Diàleg per reprendre sessió

11.6 Colorimetria targetes

Cada targeta del panell d'errors utilitza un color de fons per indicar l'estat de revisió de l'element.

Els estats i els seus colors associats són els següents: lila suau per als elements pendents de revisió, groc per a

l'element actiu (el que s'acaba de seleccionar amb el botó "Veure'l"), verd clar per als elements corregits i blau per als elements confirmats com a correctes. Quan l'usuari prem "Veure'l", la targeta activa rep a més una vora taronja que la distingeix visualment de la resta independentment del seu estat.



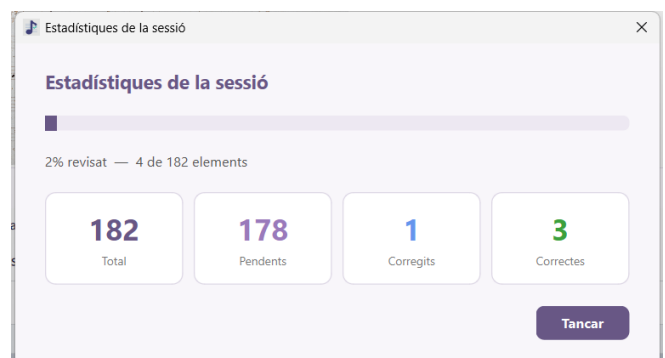
Imatge 7: Targetes d'errors amb colorimetria

11.7 Estadístiques

L'aplicació inclou un diàleg d'estadístiques que mostra l'estat global de la sessió de validació en curs.

El diàleg mostra una barra de progrés que indica el percentatge d'elements ja revisats (corregits més confirmats), i quatre targetes numèriques amb el total d'elements, els pendents, els corregits i els confirmats com a correctes.

Cada targeta utilitza el color coherent amb el del panell d'errors, de manera que l'usuari pot relacionar fàcilment els números amb els estats que ja coneix visualment.



Imatge 8: Diàleg per les estadístiques

12 CONCLUSIONS

Aquest treball ha permès dissenyar i implementar una eina funcional per a la validació interactiva de partitures OMR. L'aplicació cobreix el flux complet de validació: des de la càrrega de la imatge original i el fitxer MusicXML fins a la navegació per els errors detectats i la seva correcció directament a MuseScore, amb persistència de sessions que permet reprendre el treball en qualsevol moment.

El repte tècnic més significatiu ha estat la integració de MuseScore, donada l'absència d'una API oficial. La solució adoptada, basada en les API Win32 de Windows, ha permès no només incrustar l'editor dins de l'aplicació sinó també implementar una navegació programàtica fins a una nota concreta.

Des del punt de vista de l'enginyeria del software, el projecte ha seguit totes les activitats fonamentals: anàlisi de requisits, disseny, implementació incremental i proves. L'arquitectura de tres capes adoptada ha facilitat el desenvolupament modular i ha permès anar afegint funcionalitats progressivament sense afectar els components ja implementats.

12.1 Treball futur

Hi ha diverses funcionalitats que serien útils afegir-les en versions futures de l'eina.

Una de les més immediates seria afegir botons de navegació seqüencial al panell d'errors per evitar haver de fer scroll, i la possibilitat d'afegir notes de text lliure a cada element per distingir tipus d'incidències.

Una línia especialment interessant seria extreure estadístiques dels tipus d'errors més freqüents que genera cada sistema OMR: quines notes es confonen més sovint, en quins compassos es concentren els errors o quines figures rítmiques presenten més problemes. Això permetria identificar patrons de fallada sistemàtics i orientar millores futures en els propis motors de reconeixement, més enllà de la validació cas per cas.

Altres opcions de millora serien explorar una alternativa multiplataforma basada en Verovio que eliminaria la dependència de Windows, tot i que requeriria reimplementar les funcionalitats d'edició que ara aporta MuseScore de manera nativa.

ÚS DE LA IA GENERATIVA

En l'elaboració d'aquest treball he utilitzat la ia generativa com a eina de suport en diferents moments del procés.

Per una banda, m'ha ajudat a millorar la redacció dels textos, corregir errades ortogràfiques i gramaticals i estructurar les idees de manera més clara.

Per altra banda, també l'he utilitzat com a suport en la generació i revisió de fragments de codi, especialment en la implementació de la integració de MuseScore amb les API Win32.

En tots els casos he revisat, adaptat i validat el resultat abans d'incorporar-lo al projecte. La IA ha actuat en tot moment com un recurs de consulta i suport, no com a substitut del meu propi criteri.

AGRAÏMENTS

En primer lloc, vull expressar el meu agraïment a la meva tutora Alicia Fornes, pel seu suport i orientació durant el desenvolupament del projecte. També agrair al Pau Torras la seva ajuda i suggerències en diversos aspectes del projecte.

Per altra banda, a l'Edgar Zamora, per haver pogut compartir part del procés junts.

Per acabar, a la meva família, companys i amics pel constant suport rebut durant tot el procés.

BIBLIOGRAFIA

- [1] MusicXML. Music in MusicXML Format. [en línia] Disponible a: <https://www.musicxml.com/music-in-musicxml/>
- [2] Audiveris. Open-Source Optical Music Recognition. [en línia] Disponible a: <https://github.com/Audiveris/audiveris>
- [3] Optical Music Recognition. Optical Music Recognition: State of the Art and Major Challenges. [en línia] Disponible a: <https://arxiv.org/abs/2006.07885>
- [4] Humdrum. The Humdrum Toolkit. [en línia] Disponible a: <https://www.humdrum.org/>
- [5] KernPy. KernPy Library Documentation. [en línia] Disponible a: <https://pypi.org/project/kernpy/>
- [6] music21. music21 Library Documentation. [en línia] Disponible a: <https://pypi.org/project/music21/>
- [7] Verovio. A music notation engraving library. [en línia] Disponible a: <https://www.verovio.org/index.xhtml>
- [8] MuseScore. MuseScore Studio. [en línia] Disponible a: <https://musescore.org/en>
- [9] PyQt6. PyQt6 Library Documentation. [en línia] Disponible a: <https://pypi.org/project/PyQt6/>
- [10] SQLite. SQLite Documentation. [en línia] Disponible a: <https://sqlite.org/docs.html>