A logo of a coat of arms

Description automatically generatedUNIVERSIDAD DE BURGOS

Escuela Politécnica SuperiorA logo of a machine

Description automatically generated

Gº en Ingeniería Informática

A black and white logo

Description automatically generated

**TFG Ingeniería Informática:**

**MelodyMatrix**

Presentado por Martín González Saiz

en Burgos el X julio de 2024

Tutores D. José Manuel Galán Ordax

D. José Manuel Galán Ordax

**Resumen**

Muchas personas e instituciones dedican su vida al estudio y preservación de la música de diferentes etapas de la historia, especialmente de épocas muy lejanas. A partir de esto, el trabajo de muchas de estas personas es mantener estas obras e información, transcribiendo su contenido a un formato y material con mayor usabilidad y menor riesgo de deterioro.

Por ello existen las herramientas OMR, que es una tecnología capaz de realizar el proceso de transcripción de manera automática y segura, analizando las partituras desde cualquier formato al digital, garantizando así totalmente su preservación.

A raíz de esto, se planteo este trabajo con la finalidad de integrar un software OMR en una aplicación que permitiera el fácil uso de esta tecnología para poder transcribir las partituras con gran facilidad y comodidad.

La aplicación web esta desarrollada y diseñada para ser capaz de almacenar las partituras que el usuario en cuestión registre y poder digitalizar las que este elija, existiendo también la posibilidad de procesar las partituras con funciones de procesamiento de imágenes para aquellos usuarios más experimentados en el campo. Combinando todas estas funcionalidades se podría alcanzar la transcripción total de la partitura.

**Descriptores**

Registro de partituras, almacenamiento de partituras, digitalización de partituras, preprocesamiento de las partituras, visualización de las partituras digitalizadas.

***Abstract***

*Many individuals and institutions dedicate their lives to studying and preserving music from different historical periods, especially from distant epochs. As a result, much of their work involves maintaining these works and information by transcribing their content into formats and materials that are more usable and less prone to deterioration.*

*For this reason, Optical Music Recognition (OMR) tools exist. OMR is a technology capable of automatically and securely transcribing sheet music from any format to digital, thereby ensuring its complete preservation.*

*Considering this, this project was conceived to integrate a OMR software into an application that would facilitate easy use of this technology for transcribing sheet music with great ease and convenience.*

*The web application is developed and designed to store the sheet music registered by the user and digitize those selected by them. It also offers advanced image processing functions for users experienced in the field. By combining all these functionalities, complete transcription of the sheet music can be achieved.*

***Keywords***

*Sheet music registration, sheet music storage, sheet music digitization, sheet music preprocessing, visualization of digitized sheet music.*

**Índice General**

# 

[Índice General 2](#_Toc170502035)

[Índice de Ilustraciones 4](#_Toc170502037)

[A. Introducción 5](#_Toc170502038)

[1.1 Estructura de la memoria 5](#_Toc170502039)

[1.2 Enlaces adicionales 6](#_Toc170502040)

[B. Objetivos del proyecto 7](#_Toc170502041)

[2.1 Objetivos generales 7](#_Toc170502042)

[2.2 Objetivos técnicos 7](#_Toc170502043)

[2.3 Objetivos personales 8](#_Toc170502044)

[C. Conceptos teóricos 9](#_Toc170502045)

[3.1 Reconocimiento Óptico de Música 9](#_Toc170502046)

[3.2 Procesamiento de imágenes 10](#_Toc170502047)

[3.3 Aprendizaje automático 12](#_Toc170502048)

[3.4 Formato MusicXML 13](#_Toc170502049)

[D. Técnicas y Herramientas 15](#_Toc170502050)

[4.1 Metodología ágil – Scrum 15](#_Toc170502051)

[4.2 Herramienta de control de versiones 15](#_Toc170502052)

[4.3 Herramienta de gestión del proyecto 15](#_Toc170502053)

[4.5 Herramienta para gestionar el repositorio local-remoto 16](#_Toc170502054)

[4.6 Herramienta para la gestión de referencias bibliográficas 16](#_Toc170502055)

[4.7 Lenguaje de programación 16](#_Toc170502056)

[4.8 Interfaz gráfica 16](#_Toc170502057)

[4.9 Herramienta para la interfaz 17](#_Toc170502058)

[4.10 Herramienta de base de datos 17](#_Toc170502059)

[4.11 Herramienta de almacenamiento de archivos 17](#_Toc170502060)

[4.12 Herramienta de encriptación de contraseñas 17](#_Toc170502061)

[4.13 Codificación en ASCII 18](#_Toc170502062)

[4.14 Herramienta de conversión de PDF a imagen 18](#_Toc170502063)

[4.15 Herramientas de procesamiento de imágenes 18](#_Toc170502064)

[4.16 Herramienta de digitalización de partituras 18](#_Toc170502065)

[4.17 Herramienta de visualización de partituras 19](#_Toc170502066)

[4.18 Herramienta de despliegue con contenedores 19](#_Toc170502067)

[4.19 Herramienta para albergar la aplicación 19](#_Toc170502068)

[E. Aspectos relevantes de desarrollo del proyecto 21](#_Toc170502069)

[5.2 Metodologías 21](#_Toc170502070)

[5.3 Formación 22](#_Toc170502071)

[5.4 Audiveris 24](#_Toc170502072)

[5.5 Docker 25](#_Toc170502073)

[5.6 Problemas derivados del código 27](#_Toc170502074)

[F. Trabajos relacionados 29](#_Toc170502075)

[6.1 Audiveris 29](#_Toc170502076)

[6.2 PhotoScore 29](#_Toc170502077)

[G. Conclusiones y líneas de trabajo futuras 31](#_Toc170502078)

[7.1 Conclusiones 31](#_Toc170502079)

[7.2 Líneas de trabajo futuras 32](#_Toc170502080)

[H. Objetivos de Desarrollo Sostenible 34](#_Toc170502081)

[Bibliografía 35](#_Toc170502084)

**Índice de Ilustraciones**

[Figura 1 Reconocimiento Óptico de Música 9](#_Toc170511841)

[Figura 2 Partitura escaneada 10](#_Toc170511842)

[Figura 3 Partitura binarizada 11](#_Toc170511843)

[Figura 4 Filtro Gaussiano 11](#_Toc170511844)

[Figura 5 Partitura a erosionar 12](#_Toc170511845)

[Figura 6 Partitura erosionada 12](#_Toc170511846)

[Figura 7 Partitura a dilatar 13](#_Toc170511847)

[Figura 8 Partitura dilatada 13](#_Toc170511848)

[Figura 9 Do en clave Sol 14](#_Toc170511849)

[Figura 10 Estructura de Firestore Database 23](#_Toc170511850)

**A. Introducción**

En el sector del arte y la cultura, sobre todo en campos de la preservación y el acceso al contenido antiguo existe una gran necesidad por garantizar que las generaciones venideras puedan trabajar y estudiar estas obras. Este contenido puede corresponder a un amplio abanico de posibilidades, y dentro del sector musical puede tomar, desde partituras y libros hasta instrumentos y grabaciones. Por ello la digitalización de este tipo de material es esencial, no solo para evitar su deterioro por el paso del tiempo y su perdida debido a ello, garantizando su preservación, sino también para facilitar su acceso mediante el uso de plataformas digitales. Gracias a esto, sería posible lograr que toda esta riqueza cultural, artística y educativa, que nace de todas estas obras, pueda ser accesible con grata facilidad y eficiencia además de segura para todo el mundo.

En el caso de las partituras, el manejo, estudio, almacenamiento e investigación recae sobre muchas personas dedicadas a diferentes sectores dentro del mundo de la música, como musicólogos, músicos y educadores, los cuales se dedican y ganan la vida haciendo uso de este material. La transcripción de las partituras siempre se ha realizado manualmente, provocando que, muchas veces, se produzcan errores humanos y que el proceso sea lento e ineficiente, sobre todo con transcripciones de grandes colecciones o con partituras notoriamente complejas.

Con el gran avance de la tecnología, se han desarrollado varias herramientas y tecnologías que agilizan y optimizan este proceso. Una de las herramientas más utilizadas es el Reconocimiento Óptico de Música, la cual realiza la digitalización de partituras de manera automática. Una de las herramientas más populares y con mayor efectividad es Audiveris.

Este proyecto se enfoca en desarrollar una aplicación web que permita el almacenamiento, gestión y digitalización de las partituras de forma que sea simple y accesible de usar por el usuario. La funcionalidad principal es transcribir estas partituras desde cualquier formato hacia MusicXML, formato universal para el contenido musical digital. Tras ello se podría editar las notas a su gusto para luego descargar la nueva versión.

Otra finalidad de la herramienta es reducir la necesidad de intervención manual y generación de errores en la transcripción y edición. De esta manera, el patrimonio musical podría perdurar con mayor facilidad y el gremio de la música crecer nutriéndose de herramientas como esta.

1. **Estructura de la memoria**

A continuación, se va a presentar lo que será la estructura de la memoria con cada uno de sus apartados:

* **Introducción:** En este apartado se va a describir el contexto del proyecto y su planificación.
* **Objetivos del proyecto:** Explicación de los objetivos que tiene el proyecto, objetivos generales, técnicos y personales.
* **Conceptos teóricos:** Explica cada aspecto y la teoría aplicada al proyecto en profundidad.
* **Técnicas y herramientas:** Se listan todas las herramientas y técnicas utilizadas en el proyecto, explicando su función y adaptación al proyecto.
* **Aspectos relevantes del desarrollo:** Explicación del flujo de desarrollo del proyecto, recalcando los puntos más clave de este.
* **Trabajos relacionados:** Trabajos en los que hay cierta relación con el proyecto.
* **Conclusiones y líneas de trabajo futuras:** Conclusiones sacadas del proyecto final y aspectos que se pueden mejorar o trabajo que se puede desarrollar en base al proyecto.
* **Objetivos de desarrollo sostenible:** un listado de objetivos de desarrollo sostenible que se han alcanzado con este proyecto.

Estructura del documento de anexos:

* **Planificación del proyecto:**
* **Especificación de requisitos:**
* **Especificación de diseño:**
* **Manual del programador:**
* **Manual de usuario:**

1. **Enlaces adicionales**

En este apartado se van a adjuntar los enlaces correspondientes a las herramientas externas de control del proyecto utilizadas:

* **Repositorio de GitHub**:  
  <https://github.com/martingonzsaiz/Digitalizacion-Partituras>
* **Tablas de la gestión del proyecto de Trello**:
  + Sprint 1: <https://trello.com/b/s2npoqgv/tfg-digitalizacion-de-partituras-sprint-1-15-02-2024-01-03-2024>
  + Sprint 2: <https://trello.com/b/SEnQJM6h/tfg-digitalizacion-de-partituras-sprint-2-01-03-2024-17-03-2024>
  + Sprint 3: <https://trello.com/b/5qrTSteY/tfg-digitalizacion-de-partituras-sprint-3-18-03-2024-15-04-2024>
  + Sprint 4: <https://trello.com/b/W4oJQ7Ah/tfg-digitalizacion-de-partituras-sprint-4-15-04-2024>
  + Sprint 5: <https://trello.com/b/XWatG1p6/tfg-digitalizacion-de-partituras-sprint-5-23-05-2024-20-06-2024>
  + Sprint 6: <https://trello.com/b/OuDMWHQl/tfg-digitalizacion-de-partituras-sprint-6-21-06-2024>

**B. Objetivos del proyecto**

En primer lugar, se van a mostrar los objetivos principales del proyecto, abarcando desde los más teóricos hasta los más prácticos. También se van a abordar aspectos de desarrollo más individuales de desarrollo personal.

El proyecto principalmente se basa en proporcionar a usuarios involucrados en el mundo de la música una plataforma donde pueda realizar una gestión cómoda de su material. Pudiendo estos realizar diferentes acciones ya sea de almacenaje, digitalización, preprocesamiento, borrado, etc.

1. **Objetivos generales**

A continuación, se van a listar todos los objetivos principales del proyecto:

* Incorporar una autenticación segura y efectiva con el uso de una encriptación manejada por bcrypt, así como una gestión de usuarios sólida.
* Dar un servicio de almacenamiento de partituras online para cualquier usuario con una interacción cómoda y rápida mediante las funcionalidades de subida y borrado de partituras, a través del uso de los servicios de Firebase
* Implementar un proceso de transcripción de partituras eficiente a través del uso del software de reconocimiento óptico de música (OMR), Audiveris. A su vez, contrastar los resultados a través de un análisis de los logs generados por el proceso de digitalización de Audiveris para informar al usuario.
* Proporcionar un sistema de almacenamiento independiente para las partituras que han sido digitalizadas.
* Implementar una funcionalidad de eliminación de partituras tanto subidas sin digitalizar, como partituras digitalizadas.
* Implementar una opción de preprocesamiento manual mediante el uso de tecnologías de procesamiento de imágenes, como desenfoque o umbralización adaptativa, con el fin de mejorar la calidad de la imagen subida y digitalizable.
* Implementación de la posibilidad de visualización de las partituras digitalizadas mediante el uso de la librería Verovio.
* Implementación de una funcionalidad que permita al usuario editar las partituras
* Obtención automática de metadatos
* Utilización de servicios externos populares en el mundo del desarrollo web como las herramientas de Firebase, el Docker y sus contenedores o el despliegue con Heroku.

1. **Objetivos técnicos**

* Utilizar la metodología Scrum para el desarrollo correcto del proyecto.
* Uso de la herramienta Trello para la gestión de las tareas y avances del proyecto.
* Manejo de GitHub como herramienta para gestionar los cambios y las versiones del proyecto.
* Utilizar GitHub Desktop para el adecuado control de los cambios de la aplicación y el acceso cómodo al historial de cambios.
* Uso de Flask como framework para el desarrollo web de mi aplicación de digitalización.
* Utilizar una herramienta de manejo y gestión de referencias para la documentación del proyecto.
* Utilizar librerías de Python para el desarrollo de funcionalidades de la aplicación.
* Implementación de código que permite utilizar softwares como mini servicios en segundo plano.

1. **Objetivos personales**

* Desarrollar mi primera aplicación web en solitario
* Uso de Python como lenguaje de programación para el desarrollo
* Manejo de excepciones para el perfecto control del flujo de la aplicación y pruebas
* Manejo de la funcionalidad logging para controlar el flujo de la aplicación y solventar errores en las funcionalidades de la aplicación.
* Utilizar herramientas y servicios externos relacionados con el mundo del desarrollo web.
* Desplegar la aplicación públicamente para posibilitar su uso para cualquier usuario.

**C. Conceptos teóricos**

A continuación, se van a explicar y detallar todos aquellos conceptos y aspectos del proyecto que tengan un fundamento teórico detrás. Con ello, se podrá llegar a un mejor entendimiento del propósito del proyecto.

1. **Reconocimiento Óptico de Música**

Es una tecnología desarrollada para conseguir el reconocimiento automático y computacional de notación musical en diferentes archivos y documentos. Con ello se consigue que la partitura en cuestión pueda ser convertida a un archivo digital interpretable por la máquina. Estos archivos pueden ser de diferentes formatos, como MIDI o MusicXML con los cuales se puede manipular y trabajar con esta información musical. (1)

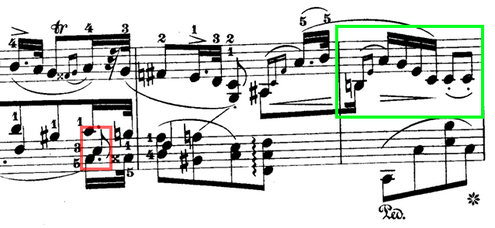


Figura 1 Reconocimiento Óptico de Música

(1)

Las OMR tienen fundamentos que se relacionan mucho con los de las tecnologías de reconocimiento óptico de caracteres, pero la realidad es que hay diferencias muy grandes: (1)

* En la notación musical es necesario registrar una serie de características bien definidas para poder diferenciar los distintos elementos. Siendo así que la tecnología tiene que ser capaz de detectar atributos como la posición de la nota o su morfología y configuración. Mientras que en el reconocimiento de caracteres está basado meramente en morfología.
* Otra diferencia notoria es que las OMR se ven obligadas a hacer un reconocimiento de la semántica para conocer aspectos como la posición vertical, añadiendo complejidad al proceso. A esto se le pude sumar que tiene que ser capaz de interpretar cualquier tipo de conjunto caracteres, ya que los elementos pueden ser desde simplemente un punto hasta elementos que abarcan prácticamente la partitura entera, además de que muchos símbolos están ligados entre ellos.

Por todo esto, a este tipo de reconocimiento se le plantea una serie de desafíos a cumplir que para alcanzarlos debe seguir los siguientes pasos:

* Lo primero es reconocer patrones en los distintos elementos de las partituras. De esta manera sería capaz de agrupar distintos símbolos musicales y mejorar el rendimiento. Para ello es crucial primero eliminar las líneas del pentagrama para llegar a una mejor detección de estos símbolos.
* También es necesario el uso de funciones de procesamiento de imágenes que ayudan a eliminar características propias de la baja calidad de partituras escritas a mano o escaneos que limitan mucho el proceso de reconocimiento.
* Por último, llegar a una extracción de metadatos que permita registrar características de la partitura como el autor, titulo, etc.

Para alcanzar todas estas metas las tecnologías de reconocimiento óptico de música utilizan aprendizaje automático, así como funciones de procesamiento de imágenes, las cuales se van a explicar en el siguiente punto.

1. **Procesamiento de imágenes**

Como se ha explicado previamente las tecnologías OMR hacen uso de funciones de procesamiento de imágenes para conseguir que la partitura llegue a un estado y características que permitan una mejor detección de sus elementos. Este proceso es necesario debido a que muchas partituras tienen una calidad pobre debido a diferentes circunstancias. A continuación, vamos a ver pasos de procesamiento de imágenes más comunes en las OMRs.

**Binarización**

Con este proceso se busca transformar la imagen digital generalmente a blanco y negro, preservando las propiedades más importantes de la imagen y facilitando la detección de los símbolos. En el proceso de procesamiento de imágenes de las OMR es importante ya que ayuda a simplificar la imagen y muchas veces reduce el exceso de información. Para llegar a un proceso de binarización generalmente se utiliza la función de umbralización descrita a continuación. (2)

* Umbralización: Primero se establece un valor para un umbral. A partir de este umbral se seleccionan solo aquellos pixeles que tienen un valor de intensidad superior al definido para el umbral, estos pixeles seleccionados se convertirán al blanco, mientras que los que no sobrepasen este valor se quedarán con un color negro.

Siendo E la imagen de entrada, f una función, T el valor del umbral y S la imagen resultante (3):

A sheet of music with notes

Description automatically generated

Figura 2 Partitura escaneada

A sheet of music with notes

Description automatically generated

Figura 3 Partitura binarizada

**Eliminación de ruido**

Esta función ayuda a eliminar la información y elementos que no son necesarios en la digitalización o que son datos no deseados en la notación musical. Esto generalmente suele ser cualquier tipo de imperfección en un documento escaneado o fotografiado, por ejemplo, manchas de polvo, dobleces de papeles, etc. Las técnicas utilizadas en este proceso son las siguientes: (4)

* Filtro Gaussiano: Es una función matemática para suavizar la imagen. Este filtro es generalmente utilizado para reducir el ruido de una imagen. En las tecnologías OMR es utilizado para reducir el ruido de fondo que suelen tener las partituras escaneadas o fotografiadas, facilitando enormemente la detección de los símbolos y líneas de los pentagramas. Siendo x e y las distancias desde el punto central del kernel y σ la desviación estándar que controla cuánto se suaviza la imagen: (4)

A sheet of music with notes

Description automatically generated

Figura 4 Filtro Gaussiano

* Filtro de mediana: Este filtro es utilizado para reducir el ruido “sal y pimienta”, sin afectar demasiado la nitidez de la imagen. Esta imperfección suele ocurrir en imágenes con pixeles de distintas tonalidades de color y muy intensas. Este al igual que en el filtro anterior es ideal para partituras escaneadas o fotografiadas donde hay desgaste de papel o un escaneo de mala calidad. (5)
* Erosión: La erosión es un tipo de operación morfológica para procesar imágenes. En el caso de la erosión, su función es la de eliminar los píxeles de los bordes de los objetos y elementos, es decir, adelgazar o reducirlos. Se suele utilizar para eliminar imperfecciones como elementos que se juntan por tener líneas muy gruesas. Para adelgazar los objetos y elementos, esta función utiliza un elemento estructurante o plantilla que se encarga de examinar cada píxel para evaluar si debe ser vaciado o no en función del estado de los pixeles de su entorno. Esto se evalúa en términos matemáticos, utilizando valores binarios, se analiza si los valores de los pixeles del entorno tienen un valor de 0 o no, si se da el caso de que alguno de los pixeles es 0, se le asigna un valor de 0 al pixel central (6). Siendo A un conjunto de pixeles de la imagen original y B el elemento estructurante, la erosión se define: (7)

(7)

Donde z son los píxeles de la imagen y indica la translación de B por z. (7).

En el contexto de las partituras puede ocurrir que haya notas que se juntan demasiado con las líneas del pentagrama o cualquier tipo de símbolo fusionándose con otro debido a una mala calidad de escaneo, dificultando la distinción de estos elementos.

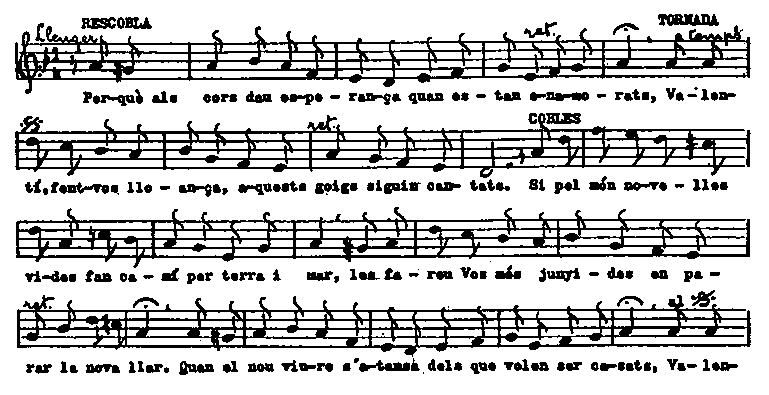


Figura 5 Partitura a erosionar

A sheet of music with notes

Description automatically generated

Figura 6 Partitura erosionada

* Dilatación: es el proceso totalmente contrario a la erosión, en lugar de adelgazar el trazado de los elementos los expande. La finalidad de esta función es rellenar los huecos y agujeros vacíos que quedan en los elementos de la imagen debido a imperfecciones de escaneo o fotografía, al igual que pasaba en el otro caso. En este caso el elemento estructurante en lugar de tomar el valor mínimo de todos los pixeles del entorno toma el máximo. Establece un valor de 1 al pixel en cuestión si el resto de los pixeles de su entorno toman un valor de 1 (6). Siendo los conjuntos A y B como en el caso anterior:

(7)

Se añaden a A aquellos píxeles z para los cuales la traslación de B intercepta A. (7)

Al igual que antes este proceso es ideal para evitar que las imperfecciones de las partituras escaneadas o fotografiadas interfieran en el proceso de digitalización. En este caso las imperfecciones pueden ser huecos en notas como por ejemplo en una negra, que al tener un hueco pueda parecer una blanca, modificando el tempo de una canción.

A sheet of music with notes

Description automatically generated

Figura 7 Partitura a dilatar

A black and white sheet of music

Description automatically generated

Figura 8 Partitura dilatada

1. **Aprendizaje automático**

Encadenando con los puntos anteriores, otro aspecto muy importante de las tecnologías OMR es el aprendizaje automático que usan para mejorar su precisión. El objetivo principal al usar aprendizaje automático es entrenar modelos que sean capaces de analizar partituras y sacar patrones comunes y complejos de la estructura de ellas (8). Las etapas que resumen el proceso completo es el siguiente:

* Entrenamiento: Se busca una estructura de datos que se pueda aplicar al entrenamiento del modelo. Tras ello se validan los modelos para llegar a un ajuste perfecto y que estos sean capaces de generalizar de forma efectiva las partituras nuevas (8).
* Optimizar los modelos: Se seleccionan las estructuras y algoritmos idóneos para optimizar al máximo las tecnologías OMR (8).
* Mejora de los modelos: Entender el funcionamiento y comportamiento de cada modelo para buscar parámetros y opciones que ayuden a mejorar su toma de decisiones para clasificar y construir las estructuras musicales (8).

Las técnicas que utilizan están enfocadas en estructurar y clasificar la música según los resultados de sus técnicas.

**Clasificación de símbolos**

Se utilizan algoritmos de redes neuronales (CNN) que trabajan sobre datos con etiquetas que referencian a imágenes de los símbolos musicales. Durante el proceso es necesario ir ajustando los parámetros del modelo para ir reduciendo el error en la predicción y detección del grupo de símbolo en cuestión. Se realiza un aprendizaje profundo para extraer de manera automática las características mas relevantes de las imágenes de los símbolos, para una clasificación precisa. (8)

**Estructura musical**

Para la construcción de estructuras musicales se utilizan una serie de aspectos:

* Modelado secuencial: Se utilizan redes para manejar las secuencias de datos, con ello los modelos aprenden patrones temporales y secuencias de música de cualquier tipo. De esta manera pueden predecir la duración de las notas, el ritmo y toda su estructura.
* Generación del formato digital: La estructura que se construye es utilizada para generar el archivo que se exportara tras la digitalización. Este archivo puede ser un formato MusicXML o MIDI, los cuales codifican las estructuras y detalles del pentagrama permitiendo incluso poder llegar a reproducir la partitura.

1. **Formato MusicXML**

Es un formato abierto y universal que recoge toda la información relacionada con una partitura. Por ello su planteamiento inicial fue que fuera una herramienta de representación de este elemento para poder ser usada en distintos programas o softwares, intercambiando esta información. El alcance de este elemento permite que el conjunto de datos sea comprensible tanto por estas herramientas dedicadas a este formato, como por personas reales, siendo posible su interpretación física. (9)

Como bien se ha dicho, una de las características y ventajas más notorias de este formato es que aporta universalidad a un tipo de dato relacionado con las partituras para poder trabajar con él con distintas aplicaciones y herramientas. Otra característica muy notoria es que permite la representación de cualquier elemento propio de una partitura, ya sea una nota, acorde, tonalidad, etc, llegando a una interpretación total de toda una compleja estructura de notación musical. (9)

Sin embargo, la característica más interesante de este formato es su estructura XML, agrupando los datos de las partituras, hace que sea interpretable tanto por humanos como por maquinas. Estos archivos se estructuran mediante secciones que especifican cada elemento de la partitura en cuestión, codificando cada aspecto del documento. En el siguiente ejemplo se muestra una imagen de una partitura de una sola nota y su representación en un archivo MusicXML. (9)

A black background with a black square

Description automatically generated with medium confidence

Figura 9 Do en clave Sol

(9)

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<!DOCTYPE score-partwise PUBLIC

"-//Recordare//DTD MusicXML 2.0 Partwise//EN"

"http://www.musicxml.org/dtds/partwise.dtd">

**<score-partwise** version="2.0"**>**

**<part-list>**

**<score-part** id="P1"**>**

**<part-name>**Music**</part-name>**

**</score-part>**

**</part-list>**

**<part** id="P1"**>**

**<measure** number="1"**>**

**<attributes>**

**<divisions>**1**</divisions>**

**<key>**

**<fifths>**0**</fifths>**

**</key>**

**<time>**

**<beats>**4**</beats>**

**<beat-type>**4**</beat-type>**

**</time>**

**<clef>**

**<sign>**G**</sign>**

**<line>**2**</line>**

**</clef>**

**</attributes>**

**<note>**

**<pitch>**

**<step>**C**</step>**

**<octave>**4**</octave>**

**</pitch>**

**<duration>**4**</duration>**

**<type>**whole**</type>**

**</note>**

**</measure>**

**</part>**

**</score-partwise>**

(9)

**D. Técnicas y Herramientas**

En esta sección de la memoria se va a listar y describir cada herramienta utilizada en el proyecto. También se van a citar aquellas que se hayan valorado usar para el desarrollo del proyecto o hayan estado presentes en alguna etapa del proyecto, pero se hayan descartado, explicando su motivo.

1. **Metodología ágil – Scrum**

La metodología SCRUM se basa en el principio ágil de desarrollo iterativo e incremental.

Con esta º estado de cada tarea. Cada sprint comienza y termina con una reunión con el tutor, donde se discuten y tratan diferentes aspectos del proyecto.

1. **Herramienta de control de versiones**

* **Herramienta:** GitHub

GitHub es una herramienta de control de versiones utilizada para gestionar y controlar los cambios realizados en código de forma segura. A su vez permite a los desarrolladores coordinarse en un mismo proyecto contribuyendo a el código mediante la integración de cambios. También es la mayor red de código abierto donde multitud de proyectos están publicados.

Principalmente se ha utilizado esta herramienta por su popularidad y fácil uso, además de para tener un control seguro de los cambios que se han realizado.

1. **Herramienta de gestión del proyecto**

* **Herramienta:** Trello

En este proyecto se ha utilizado Trello para la planificación de las tareas, debido a su facilidad de uso y gratuidad.

1. **Herramienta para realización de la documentación**

* **Herramienta:** Microsoft Word

Se valoró entre el uso de LaTeX o Microsoft Word, pero finalmente se optó por este último dado su fácil uso y conocimiento de sus herramientas. LaTex requiere de un estudio y conocimiento especifico más enfocado en investigaciones científicas y matemáticas que para este proyecto no es del todo necesaria.

1. **Herramienta para gestionar el repositorio local-remoto**

* **Herramienta:** GitHub Desktop

Durante el desarrollo del proyecto se han registrado y gestionado los cambios en el repositorio del proyecto con GitHub Desktop. Esta herramienta permite la interacción rápida con las funcionalidades de GitHub mediante interfaz de usuario para realizar las acciones de Git, facilitando el proceso y haciendo el entorno más cómodo que con el uso de comandos en la línea de comandos. Además, ofrece una minisección con el historial de cambios realizados que ayuda al usuario a visualizar el orden del desarrollo del proyecto.

1. **Herramienta para la gestión de referencias bibliográficas**

* **Herramienta:** Zotero

Zotero es un gestor de referencias bibliográficas que ayuda a recoger y gestionar la documentación e información usada en una investigación o desarrollo.

Se ha utilizado en el proyecto ya que es libre y gratuito y permite insertar las referencias y objetos con gran facilidad en documentos de texto como Microsoft Word. (10)

**Alternativa**: En un inicio se empezó el proyecto contando con Mendeley como gestor de referencias, pero se migró a Zotero ya que el primero tiene un enfoque más comercial además de que limita sus funcionalidades a la versión gratuita, mientras que Zotero es gratis y de código abierto.

1. **Lenguaje de programación**

* **Herramienta:** Python

En este proyecto se ha utilizado Python debido a que este lenguaje simplifica mucho la gestión de dependencias y entornos gracias al gestor de paquetes pip. Además, es ideal para crear aplicaciones rápidamente haciendo cualquier tipo de prototipo. Otra característica que recalcar es que permite trabajar con el patrón MVC para diferenciar claramente estos tres conceptos en la estructura de los datos. (11)

1. **Interfaz gráfica**

* **Herramienta:** Flask

Flask es un micro Framework que ayuda a reducir el trabajo en el desarrollo y creación de aplicaciones web. Se les da este nombre porque desde un inicio tienen lo esencial para poder empezar a desarrollar una aplicación web. Además, su característica principal es que su funcionalidad se puede ampliar a gusto del usuario mediante plugins y librerías. (12)

**Alternativa:** Se ha preferido Flask frente a Django ya que la curva de aprendizaje del primero frente al segundo es notablemente menor, haciendo que el desarrollo sea más ágil. Además, en principio no se requerían todas las funcionalidades que incluye Django para el proyecto, pudiendo ir incluyendo las necesarias mediante librerías en Flask como solución. (13)

1. **Herramienta para la interfaz**

* **Herramienta:** BootStrap

Bootstrap es un framework de front-end muy popular que combina CSS con JavaScript para dar estilo a diferentes elementos de las vistas HTML de interfaces web. Esto ayuda a facilitar la interacción y comunicación con el usuario mediante menús de navegación, botones, formularios, tarjetas, modales y barras de navegación. (14)

Se ha decidido utilizar esta herramienta debido a que no se ha considerado relevante para el proyecto estilizar y maquetar el enfoque visual de la aplicación mediante la creación de fichero CSS y planificación de patrones de diseño.

1. **Herramienta para la base de datos**

* **Herramienta:** Cloud Firestore

Cloud Firestore es una base de datos NoSQL dedicada al almacenaje y gestión de los datos de manera dinámica y fácil. Este servicio utiliza documentos y colecciones como estructura de datos jerarquizando estos elementos para relacionarlos y realizar las consultas con facilidad. Además, destaca por escalar automáticamente en función del tamaño del conjunto de resultados. (15)

**Alternativa:** Inicialmente, el proyecto utilizaba MySQL, gestionado a través de phpMyAdmin y el paquete XAMPP. Sin embargo, se decidió migrar a Firebase buscando sus ventajas, como el escalado automático y un fácil uso. Mientras que la primera opción requería migraciones cada vez que se modificaban los modelos, lo que hacía el proceso más tedioso y menos eficiente.

**4.11 Herramienta de almacenamiento de archivos**

* **Herramienta:** Google Cloud Storage

Esta herramienta es un servicio de almacenaje de objetos en la nube pudiendo estos ser de cualquier tipo y cantidad, siendo posible acceder a estos datos en un momento deseado. El almacenaje se lleva a cabo mediante buckets que son contenedores donde se almacena cada dato para luego ser identificado con facilidad. (16)

En mi proyecto se han utilizado los buckets para almacenar las partituras, tanto procesadas como sin procesar.

**4.12 Herramienta de encriptación de contraseñas**

* **Herramienta:** Bcrypt

Es una biblioteca que se utiliza para aumentar la seguridad de las aplicaciones haciendo uso de un hasheo intensivo de las credenciales evitando que usuarios malintencionados tengan éxito en sus ataques a la aplicación mediante diferentes estrategias.

**4.13 Codificación en ASCII**

* **Herramienta:** Base64

Esta herramienta es utilizada para convertir datos binarios en texto ASCII.

En este proyecto aparece en códigos que utilizan esta herramienta para codificar archivos JSON de servicios externos con el propósito de mejorar la seguridad e integración con el Docker de despliegue y el desarrollo de la aplicación.

**4.14 Herramienta de conversión de PDF a imagen**

* **Herramienta:** pdf2image

Esta herramienta es biblioteca de Python para convertir PDF a imagen. En la aplicación se ha utilizado para asegurarse de que el archivo es una imagen, formato compatible para el realizar el proceso de preprocesamiento.

**4.15 Herramientas de procesamiento de imágenes**

* **Herramientas:** OpenCV y NumPy

OpenCV es una biblioteca de código abierto que tiene una amplia gama de aplicaciones las cuales pueden abarcar un amplio abanico que pueden ir desde la manipulación de imágenes hasta el reconocimiento de patrones homogéneos en imágenes y el aprendizaje automático. (17)

Numerical Python es una biblioteca utilizada para realizar operaciones matemáticas y estadísticas avanzadas. En el proyecto se ha utilizado para procesar las imágenes en las funciones que manejan la conversión y el procesamiento, alterando las matrices de las imágenes para su manipulación. (18)

**4.16 Herramienta de digitalización de partituras**

* **Herramienta:** Audiveris

Audiveris es un sotftware de reconocimiento óptico de música de código abierto. Este es utilizado para convertir partituras escaneadas en formato digital. El proceso que es muy parecido al realizado para el reconocimiento de caracteres en textos, pero adaptándolo a elementos musicales como pentagramas, notas, tempos, etc. Normalmente tras el proceso de digitalización el archivo pasa a un formato MusicXML, formato universal y compatible para su interacción con la mayoría de los programas. (19)

**Alternativa:** Al elegir un software a integrar en la aplicación para poder transcribir las partituras se valoró usar, además de Audiveris, SharpEye, Music Reader o MuseScore. Al final se optó por Audiveris porque esta herramienta ofrece resultados de transcripción superiores al resto, lo que aportaba una solución más sólida y adaptable a los intereses de los usuarios que se focalizan en la transcripción. Además Audiveris cuenta con una comunidad más amplia, lo cual ayuda a acceder a una gran variedad de recursos para cualquier tipo de desarrollo con esta herramienta, ya sean foros, repositorios, códigos, etc.

**4.17 Herramienta de visualización de partituras**

* **Herramienta:** Verovio

Es una biblioteca de software de código abierto que permite la renderización de la música. Permite un gran abanico de tipos de notación que van desde notación musical estándar hasta históricas y modernas. (20)

Se ha utilizado esta herramienta para la visualización de las partituras que han obtenido un resultado aplicable para poder ser visualizadas. Ha entrado en consideración ya que ofrece un buen servicio de representación de notaciones complejas.

**4.18 Herramienta de despliegue con contenedores**

* **Herramienta:** Docker

Es una plataforma que permite el encapsular máquinas virtuales de diferentes tipos para ser transportadas, copiadas o implementadas, ideal para el despliegue de aplicaciones en la nube. Permite encapsular aplicaciones con todas sus dependencias y librerías en este Docker lo que abre posibilidades de mejorar entornos colaborativos y de testeo gracias a esta portabilidad. Otro punto que recalcar es que asegura que la aplicación se ejecutará igual en producción que localmente al contar con este punto de aislamiento. (21)

Esta herramienta ha sido ideal para la aplicación dada la complejidad nacida del gran número de dependencias, así como de la utilización de softwares como mini servicios haciendo que la migración y gestión de la aplicación a la capa publica o la nube sea más compleja de gestionar. Por ello los contenedores han ofrecido una solución óptima al permitir aislar y encapsular cada componente con todas sus dependencias.

**4.19 Herramienta para albergar la aplicación**

* **Herramienta**: Heroku

Heroku es una herramienta que se encarga de la infraestructura de despliegue de una aplicación. Mediante el uso de contenedores virtuales, que son totalmente escalables, almacena y ejecuta las aplicaciones. Además, soporta los principales lenguajes de programación del gremio, pudiéndose instalar mediante buildpacks. (22)

Se valoró desplegar la aplicación con Heroku dado su plan gratuito y su fácil adaptación a los contenedores Docker (se explica más adelante). Al final se revertió la implementación de este servicio dado que el contenedor sobrepasó el límite de memoria que permite el plan inicial.

**E. Aspectos relevantes de desarrollo del proyecto**

A continuación, se va a citar y explicar cada punto importante en el desarrollo de la aplicación, abordando tanto las decisiones tomadas con sus razonamientos, como los bloqueos y problemas que han surgido en el proceso.

1. **Inicio del proyecto**

La idea de este proyecto nació de la idea, explicada anteriormente, de la necesidad de muchas entidades del mundo de la música por transcribir fragmentos de música, obras, partituras en peligro de deterioro o perdida por sus condiciones deplorables por el paso de tiempo. En concreto muchas universidades e institutos de musicología necesitan de recursos para la enseñanza, Por ello debe existir la posibilidad de poder preservar este material y convertirlo en un formato que sea accesible para alumnos, profesores e investigadores. Dado que hoy en día el formato más funcional de transmitir datos es digitalmente es necesario el desarrollo de tecnologías que permitan realizar esta conversión de manera efectiva.

Consulté información acerca de este proyecto a José Manuel, para valorar su alcance y posibilidades. Tras la primera reunión, habiendo explicado y aclarado los objetivos, dudas y puntos a alcanzar consideré el proyecto. Motivado por la propuesta y pasión por este arte y cultura en el mundo, acepté el trabajo y empezamos a planear los primeros pasos y preparativos del proyecto.

1. **Metodologías**

Durante el desarrollo del proyecto se ha seguido una metodología ágil de gestión y desarrollo de proyectos Scrum. Se decidió aplicar esta metodología con el fin de concentrarse en la iteración y la mejora continua del proyecto proyectos. Dado que en proyecto ha sido totalmente individualista no se ha sacado provecho del aspecto colaborativo que esta metodología aporta, pero sí en lo que a ciclos iterativos respecta. (23) Las características clave y más notorias en el desarrollo del proyecto son las siguientes:

* Reuniones en cada sprint, conectando cada sprint con ellas con el fin de hablar los avances, cambios y problemas que se han presentado en el sprint anterior y tratando los objetivos, pautas y pasos a seguir en el siguiente.
* Se ha llevado un enfoque incremental.
* Se han utilizado etiquetas para clasificar la categoría de cada tarea, dividiéndolas en: Development (Desarrollo), Documentation (Documentación), Configuration (Configuración), Investigación (Research), Discusión (Discuss) y Completado (Completed).
* Se han utilizado bloques para marcar el estado de cada tarea habiendo: Reuniones donde se especifica el momento en el que se han realizado reuniones, tareas por hacer, tareas en curso, tareas bloqueadas y tareas completadas.
* Se ha utilizado Trello para la gestión de todas estas características.
  + Cada tarea ha sido estimada por día, fijando un tiempo estimado que debe llevar o en el que se debe trabajar en esa tarea y el tiempo real que ha llevado. Se ha llevado este procedimiento para tener una visión clara del progreso y gestión del tiempo, así como para detectar la desviación de tiempos debido a errores o bloqueos.

1. **Formación**

Algunas de las herramientas utilizadas en el proyecto han requerido de una investigación y un estudio previo para poder desarrollar un trabajo con su uso. Este estudio ha podido ser desde tutoriales o listas de reproducción de youtube con guías, hasta revisar la documentación oficial de la herramienta en cuestión.

**GitHub**

A pesar de que esta herramienta es crucial y básica en el mundo de la programación, fue necesario hacer un miniestudio en el que revisar sus funcionalidades principales empezando por Git y su adaptación en la interfaz de GitHub. Finalmente, para disminuir el costo de manejar comandos por cada cambio se decidió integrar el uso de GitHub Desktop que ofrece una interfaz de usuario cómoda e intuitiva.

**Bootstrap**

Con el fin de no profundizar en tareas de diseño y maquetación, se decidió utilizar la librería Bootstrap, implementándola en todas las vistas con un diseño estándar. De esta forma se ahorró tiempo y recursos y a su vez se consiguió una interfaz de usuario válida. Para conocer la implementación necesaria en mis vistas se accedió a la documentación oficial de Bootstrap:

<https://getbootstrap.com/docs/4.1/getting-started/introduction/> (24)

**Html**

Para la visualización de las vistas y ventanas de la aplicación se han programado sencillos archivos Html con una estructura típica de base, de manera que permita la interacción con el usuario y manejo de los contenidos. Se hace uso de sintaxis de plantillas Jinja2 para el manejo de las alertas que mantienen informado al usuario, además de diferentes funcionalidades dependiendo de la vista. Para la buena gestión de los contenedores y el estilo de las vistas se ha hecho pequeñas implementaciones de CSS. Todo esto ha llevado varios ratos de búsqueda de ejemplos e implementaciones de base, así como investigación de las documentaciones correspondientes a la herramienta.

<https://www.w3schools.com/html/html_basic.asp> (25)

<https://www.w3schools.com/html/html_examples.asp> (26)

<https://jinja.palletsprojects.com/en/3.1.x/templates/> (27)

**Flask**

Este framework ha sido uno de los pilares más importantes del proyecto y más detalladamente de la aplicación. Si bien durante la carrera ya se había trabajado con Python en alguna asignatura, fue necesario conocer los aspectos y estructura del entorno para poder realizar el desarrollo correcto. Por ello se investigó en varios cursos y tutoriales para dar con el contenido más completo.

<https://www.youtube.com/watch?v=QG-RW4XpqTg&list=PLsvvBhdpMqBzPV-kZ_GfsdP_SqQcQQ68C>

**Firebase**

Firebase es una de las herramientas base de este desarrollo, utilizándose en parte de la lógica de la aplicación en varios aspectos. Varios de los servicios de Firebase son utilizados en las funcionalidades principales de la aplicación. Para conocer mejor que configuración era necesaria para el correcto funcionamiento de la aplicación con esta herramienta se consultó una serie de guías y videos que ayudaron a poner en contexto su implementación en el desarrollo. Además, también se accedió a la documentación oficial de Firebase para conocer mejor la estructura y funcionamiento de las distintas funciones.

<https://www.youtube.com/watch?v=tyF6mVR-ego> (28)

<https://www.youtube.com/watch?v=NAD90zLMRV4&t=46s> (29)

<https://community.listopro.com/aprende-a-crear-pantallas-de-login-con-flask-y-firebase/> (30)

<https://firebase.google.com/docs/reference/admin/python> (31)

Con estas guías en conocimiento e implementaciones de prueba y error se consiguió llegar a un desarrollo en el que se cumpliera diferentes características relacionadas con bases de datos y almacenamiento en la nube, encaminadas a los objetivos:

* + Autenticación y gestión de usuarios: Con el uso de Firebase se realiza un proceso de autenticación de usuarios comparando el nombre de usuario y las contraseñas. Se almacenan los datos de los nuevos usuarios en el registro con la estructura propia de Firebase organizada en documentos y colecciones.

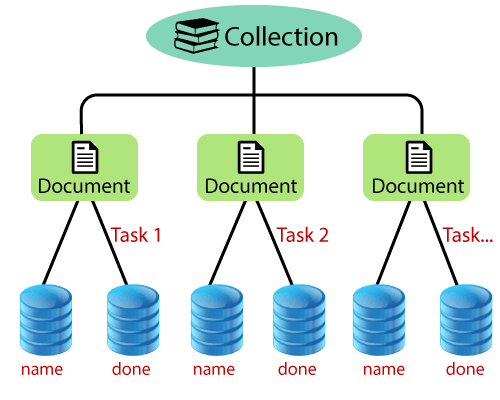


Figura 10 Estructura de Firestore Database

(32)

* + Almacenamiento y gestión de los archivos: Con el uso Firebase Storage se consigue llegar a funcionalidades de subida, descarga y eliminación de partituras. En todas estas se utilizan referencias de blob que son enlaces a los sistemas de archivos de la nube.
  + Gestión de los datos de la aplicación: Al igual que se hace con los usuarios, con las partituras se almacenan versiones de estas con sus detalles en la base de datos de Firestore con el uso de los documentos dentro de las colecciones.
  + Integración de Firebase Admin: se consigue configurar Firebase con el manejo de las credenciales y su correspondiente configuración con el fin de poder interactuar con los servicios de Firebase en el resto del código.

**OpenCV**

Es una de las técnicas ya estudiadas en algunas asignaturas de la carrera sin embargo fue necesario un repaso. En el momento de desarrollar una funcionalidad que permitiera al usuario preprocesar las partituras con el fin de conseguir resultados más legibles y visibles, incluso llegando a resultados y contenido con la digitalización de Audiveris, fue necesario documentarse acerca de las funciones e interacciones típicas del procesamiento de imágenes. Con ello se llegó a la página oficial de OpenCV la cual cuenta con una documentación muy extensa y completa con ejemplos de implementaciones que sirvieron para el desarrollo del código.

<https://opencv.org/> (33)

<https://opencv.org/get-started/> (34)

También se busco implementaciones en proyectos reales con explicaciones de usuarios reales en blogs y foros:

<https://omes-va.com/videostreaming-flask-opencv-python/> (35)

**NumPy**

A su vez también se utilizó NumPy para que OpenCV pudiera realizar las operaciones de procesamiento de imágenes. Con NumPy se proporciona una estructura de datos multidimensionales convirtiendo una imagen en un array. Se accedió a documentación oficial para ver ejemplos descriptivos de su funcionamiento.

<https://numpy.org/doc/stable/user/numpy-for-matlab-users.html> (36)

<https://www.javatpoint.com/how-to-convert-images-to-numpy-array> (37)

**Verovio**

Verovio es la librería que se encarga de la representación visual de las partituras en la vista. Cada una de las partituras es manejada en con Verovio en la propia vista a través de secciones de JavaScript. Para la implementación de este código y lógica fue necesaria una investigación completa de la documentación de Verovio en su página web.

<https://book.verovio.org/first-steps/getting-started.html> (38)

1. **Audiveris**

El aspecto mas importante de la aplicación es el uso de Audiveris en segundo plano, ya que gracias a esta implementación es posible automatizar el proceso de conversión de las partituras para su posterior manipulación en otras plataformas o actividades. Pero para llegar a este desarrollo fue crucial el estudio e investigación de la documentación oficial del software disponible en su repositorio.

<https://github.com/Audiveris/audiveris> (39)

Con la revisión de varios apartados su instalación consistió en clonar el repositorio y ejecutar el comando de construcción.

1. **‘git clone** [**https://github.com/Audiveris/audiveris.git**](https://github.com/Audiveris/audiveris.git)**’**
2. **‘cd audiveris’**
3. **‘gradlew.bat build’**

Además, gracias a toda la información contenida en este archivo, fue posible entender y conocer las posibilidades que ofrece este software. Entre todas las secciones la de mayor interés para el proyecto fue en específico la que detalla los parámetros y opciones de ejecución en interfaz de línea de comandos que tiene Audiveris. Estudiando los parámetros de la ejecución por comandos se encontró una combinación con la cuál era posible ejecutar Audiveris sin interfaz gráfica de manera que hiciera de miniservicio para la aplicación.

* **‘java -cp "Audiveris-5.3.1/lib/\*" Audiveris’**

Con esto en conocimiento el código consistió utilizar un script ejecutable desarrollado en el que se ejecutara audiveris a partir de este comando. Hubo que construir el comando teniendo en cuenta las secciones del comando se usaron rutas a los ejecutables de Java y directorios para gestionar los archivos. Como este script es usado en el Docker fue necesario adaptar rutas relativas al posterior directorio que tendrá el entorno virtual del Docker.

La ejecución de Audiveris de esta manera genera tres tipos de archivos:

* Un archivo OMR: es un tipo de archivo utilizado por Audiveris y, al igual que MusicXML este archivo es el resultado del proceso de digitalización, el cual puede ser usado por otras herramientas posteriormente. Sin embargo, este archivo a diferencia de MusicXML alberga más información relacionada con la transcripción de la partitura en cuestión. En nuestro contexto no es necesario usar este archivo. (40)
* Un archivo MusicXML: es el archivo con el que se trata en la aplicación y es el formato universal dentro del mundo de la música. Es en esencia una representación textual de la partitura que puede ser interpretado por otros softwares de música, reproductores y otros sistemas. (41)
* Un archivo Log: este archivo es el que informa de todos los eventos ocurridos en el proceso de digitalización de Audiveris. Ha resultado de utilidad para ser analizado y comprender los errores que causaban que Audiveris no pudiera digitalizar ciertas partituras. Como hay muchas situaciones en las que Audiveris no es capaz de realizar la digitalización con éxito se ha considerado implementar un código que busque ciertas cadenas de caracteres en el log y en el caso de existir informar al usuario al respecto.

1. **Docker**

El uso de un Docker ha sido otra pieza clave en el desarrollo del proyecto. Dado que en este proyecto se hace uso de softwares externos como es Audiveris, el uso de un contenedor en el que recoger todo el conjunto es ideal para poder exportar el proyecto o desplegarlo.

Lo primero antes de ponerse con la construcción del Docker fue documentarse del proceso de construcción de un Docker y de un patrón estándar o común en este tipo de proyectos a la hora de implementar el Dockerfile. Se accedió a la documentación oficial de Docker, así como a algún foro y ejemplo de implementación de un Dockerfile. Además, también se tuvo en cuenta el uso de Gunicorn para el despliegue del Docker en Heroku, siendo necesario consultar alguna implementación como ejemplo.

<https://docs.docker.com/reference/cli/docker/image/build/> (42)

<https://www.cherryservers.com/blog/docker-build-command> (43)

<https://stackoverflow.com/questions/43925487/how-to-run-gunicorn-on-docker> (44)

Para ello hubo que repasar todas las librerías y herramientas de las que hace uso la aplicación para implementar un fichero Docker que al ejecutar el comando de ejecución del Docker no diera ningún error ni problema.

* Imagen base: **‘FROM python:3.11-slim as base’** se utilizó esta versión de Python ya que es muy reciente y ligera.
* Directorio de trabajo: ‘**WORKDIR /app’** se estableció /app como directorio de trabajo en el que se ejecutarían todos los comandos del contenedor.
* Instalación de dependencias: en este punto se buscó realizar todas las instalaciones en una sola línea de comandos para optimizar el Dockerfile lo máximo posible. En primer lugar, se ejecutaría un comando para mantener actualizado el entorno para luego poder realizar todas las instalaciones de las dependencias.

**‘RUN apt-get update && apt-get install -y --no-install-recommends \**

**git \**

**curl \**

**build-essential \**

**libtesseract-dev \**

**tesseract-ocr \**

**libfreetype6-dev \**

**ghostscript \**

**imagemagick \**

**openjdk-17-jdk \**

**dos2unix \**

**poppler-utils && \**

**apt-get clean && rm -rf /var/lib/apt/lists/\*’**

* Instalación de Rust: ‘**RUN curl https://sh.rustup.rs -sSf | sh -s -- -y’** se realiza la instalación de la dependencia de Rust que es utilizada para la compilación de bycrypt que es una librería de hasheo de contraseñas, así como para la inclusión de los scripts en el contenedor.
* Construcción de Audiveris: En esta etapa se realiza la construcción de Audiveris. Como ya se ha mencionado anteriormente la instalación de Audiveris se hizo en base a la documentación encontrada en su repositorio oficial. (39) En base a los comandos de instalación y construcción se creó un comando único que realizara todo el proceso de preparación de esta herramienta para la aplicación.

**FROM base as audiveris-builder**

**RUN git clone https://github.com/Audiveris/audiveris.git && \**

**cd audiveris && \**

**./gradlew build**

* Final del Dockerfile: Tras todo esto se copiaron todos los archivos necesarios de audiveris, se crearon los directorios utilizados en la aplicación, se copió e instalo las dependencias del archivo de requerimientos creado en linea de comandos (comando pip freeze > requirements.txt) (43), y se preparó el script de ejecución de Audiveris en el que fue necesario el formato ‘dos2unix’ evitando cualquier problema con otros formatos. Por último, se expuso que el puerto fuera el 5000 además de que la aplicación se iniciara con Gunicorn.

1. **Problemas derivados del código**

En esta sección de la memoria se van a explicar las etapas que han tenido bloqueos y problemas, así como momentos en los que ha sido necesario revertir los avances en el desarrollo.

En un inicio al clonar e instalar Audiveris me encontré con problemas para ejecutar el software ya que el nombre de los ficheros y archivos no se asemejaban con exactitud a lo que viene explicado en la documentación de Audiveris. Además, el proceso de construcción del comando a través del script y su implementación en código fue muy complejo de completar llegando a ser la tarea más duradera del proyecto, llegando a un coste de tiempo muy elevado en comparación con lo que se estimaba.

El siguiente punto que añadir son los problemas con los que me encontré al instalar las dependencias para la aplicación en el Docker, obtuve muchos errores en la construcción y compilación por la compatibilidad de los paquetes y la ambigüedad en las rutas relativas, sobre todo en el directorio de Audiveris, este proceso llevo más tiempo del esperado debido a estos bloqueos. Se basó en un ciclo de prueba y error hasta dar con la versión final, la cual funcionaría. Además, este proceso se veía muy ralentizado al tener que manejar tantas versiones de contenedores e imágenes que bloqueaban el sistema debido a un colapso en el almacenamiento.

Otro aspecto que tratar son las dificultades que se presentaron al adaptar los servicios de Firebase al contexto del proyecto. Esta tecnología es muy nueva y prácticamente desconocida para mí, por lo que fue necesaria una buena investigación de esta herramienta en su documentación y páginas web.

En la implementación de Verovio apareció algún incidente con su instalación, debido a que fue necesario configurar las variables de entorno del sistema y esto bloqueo el avance del proyecto un tiempo considerable. En la implementación de una funcionalidad capaz de eliminar partituras también se presentaron dificultades debido al acceso al bucket de Firebase.

En el momento del despliegue se consideró usar Heroku para ello. Se tomaron dos caminos posibles el primero fue desplegar la aplicación haciendo uso de un archivo Procfile y buildpacks en Heroku. Esta opción dio problemas debido a que Heroku no permite la ejecución de scripts en tiempo de compilación sin configurar sus buildpacks, y ello conllevaba realizar las instalaciones necesarias de Audiveris en estos. Además, los archivos que se generaban con los scripts no persistían por alguna razón. La opción que dio resultados fue utilizar el Docker creado para tener un mejor control del entorno de ejecución y usar Heroku para configurar la aplicación y desplegarla. También dio problemas por no estar bien adaptada al contexto la inicialización de la aplicación en código y los errores en las interpretaciones del puerto en el Dockerfile al desplegar la aplicación con Heroku. El despliegue se terminó, pero la funcionalidad de digitalizar las partituras no se puede completar en la versión desplegada de Heroku ya que sobrepasa la memoria permitida en mi plan gratuito de mi cuenta.

Un punto importante que aclarar es el hecho de que Audiveris no es capaz de digitalizar todos los archivos que se suben, fallando en aquellos en los que la resolución no alcanza la permitida o siendo necesarias ciertas más funciones de preprocesamiento. Sin embargo, tras varias pruebas de procesamiento de imágenes no fue posible encontrar la manera de dar con una solución óptima. Se valoró el uso de servicios que hagan uso de modelos preentrenados de partituras de inteligencia artificial, pero tras un tiempo de investigación no se dio con ellos. Buscar una forma de entrenar un modelo que implementar en la aplicación que fuera capaz de interpretar todo tipo de partituras mejoraría la funcionalidad de la digitalización, pero el coste de tiempo era demasiado para las fechas.

Por último, probablemente el aspecto más importante de esta sección es el hecho de que no se consiguió implementar una funcionalidad en la aplicación para permitir al usuario editar las partituras digitalizadas. Esto se debe en gran parte a dos razones.

La primera es que para poder realizar esta implementación son necesarios muchos conocimientos técnicos de teoría musical, de los que se carecía. Si bien se tenía en un inicio una base de este campo no era suficiente para llegar a una implementación así. Sin embargo, no se le dio mucha importancia a esta traba ya que era posible y viable implementar una funcionalidad de edición básica. Pero el bloqueo de esta funcionalidad fue total con la segunda razón.

Para poder llegar a implementar una funcionalidad para editar las partituras eran necesarias implementaciones de funciones que manejaran Verovio y archivos MusicXML y ello implicaba llevar a cabo una investigación y estudio exhaustivo de esta herramienta, que dadas las fechas era inviable. Por ello se queda una importante funcionalidad sin completar dadas las circunstancias y las condiciones.

**F. Trabajos relacionados**

En este sector de la industria hay una gran variedad de trabajos y proyectos que implementan reconocimiento óptico de música. Estas herramientas pueden ser usadas para digitalizar las partituras o interactuar con las aplicaciones para facilitar el aprendizaje.

A continuación, se van a listar y enumerar trabajos que guardan relación con el proyecto.

* 1. **Audiveris**

En primer lugar, es importante hablar de Audiveris en profundidad ya que es el software utilizado en el proyecto. Al igual que cualquier OMR, Audiveris es capaz de convertir partituras en formato digital, con la posterior finalidad de editar o manipular este elemento. Todas las tecnologías de reconocimiento óptimo de caracteres utilizan técnicas de procesamiento de imágenes como puede ser la binarizacion o segmentación y Audiveris contiene los siguientes aspectos:

* Machine Learning: Audiveris destaca por utilizar algoritmos de aprendizaje automático para mejorar constantemente su precisión en la detección de notación musical. Sus modelos están continuamente entrenándose con grandes bases de datos relacionadas con partituras y más contenido musical.
* Generación y exportación en MusicXML: tras el proceso de transcripción y procesamiento es posible exportar el resultado en formato MusicXML.  
  1. **PhotoScore**

PhotoScore es un programa de escaneo equivalente a Audiveris y otras tecnologías OMR, también es capaz de convertir archivos PDF en notación editable y reproducible. Es capaz de leer una variedad de notaciones que pueden ir desde tablaturas de coro hasta registros preparados para orquestas. Además de convertir las partituras a MusicXML, PhotoScore ofrece más opciones de formato como MIDI o Sibelius. Esta tecnología al igual que Audiveris utiliza varias técnicas de procesamiento de imágenes. Sus características más notorias son:

* Procesamiento de imágenes
* Vision de computadora
* Machine learning
* Inteligencia artificial

Otro punto que recalcar es que implementa una técnica de procesamiento extra, la cual es la corrección de Skew que ajusta la imagen cuando el escaneo este torcido por alguna razón. (44)

**G. Conclusiones y líneas de trabajo futuras**

Por último, en esta sección se van a tratar las conclusiones sacadas del proyecto, tanto de su desarrollo, como de su planteamiento y final. Además, también se van a hablar de las líneas de trabajo futuras que serían necesarias para la aplicación actual.

1. **Conclusiones**

En general este proyecto ha sido después de todo complejo de llevar a cabo. Esto se debe sobre todo en parte a la ambigüedad, dentro del contexto del proyecto, del desarrollo que era necesario seguir para llegar al fin propuesto. En consecuencia, a esto, nació una gran duda con lo que respecta a las tecnologías e implementaciones a realizar, así como de las funcionalidades de las que debía gozar la aplicación. En muchos puntos del desarrollo apareció mucha incertidumbre por si el desarrollo estaba bien encaminado y si albergaba un sentido lógico a la hora de reflejarlo como una aplicación funcional. Además, debido a que, en un inicio gran parte de las tecnologías utilizadas eran desconocidas, muchas etapas del desarrollo se basaron en un ciclo de implementación, prueba y error, provocando que muchas veces los resultados se consiguieran “a la fuerza” o tomando una implementación que en un inicio no era del todo la esperada.

Todo ello sumado a que el desarrollo de este proyecto se ha visto condicionado por una limitación de tiempo, ha provocado que algunos de los objetivos fijados para el proyecto no hayan salido como se esperaba y que la aplicación se vea necesitada de más funcionalidades.

**Objetivos generales cumplidos**

Como objetivos generales, se consideran como cumplidos todos a excepción de los dos siguientes:

* Llegar a un desarrollo que permita al usuario editar las partituras digitalizadas, siendo capaz de manipular y alterar los elementos de la partitura a su antojo, mediante la obtención de metadatos. Esta implementación no fue posible sacarla adelante por falta de tiempo y conocimientos en su desarrollo, a pesar de que fue uno de los objetivos propuestos.
* Un objetivo que ha sido parcialmente fallido es el hecho de que la aplicación no es capaz de digitalizar todas las partituras, fallando en aquellas en las que sus condiciones y características son complejas de analizar debido a ciertos factores de procesamiento, ya sea porque la partitura tiene exceso de ruido, mucha saturación, trazados muy gruesos, baja resolución, etc.

**Objetivos técnicos cumplidos**

Como objetivo técnicos cumplidos se pueden considerar los siguientes:

* Todos aquellos relacionados con la gestión de proyectos y de tareas haciendo uso además de herramientas dedicadas a ello.
* También aquellos enfocados en el correcto manejo del repositorio de GitHub y sus interacciones con los cambios.
* Finalmente se ha utilizado Flask, realizando todas las implementaciones y desarrollo en este framework, tal y como se esperaba.
* Se ha utilizado una herramienta de gestión de referencias, Zotero, para referenciar todos los textos e información obtenida de distintas fuentes. Además, también se ha utilizado como base de datos donde guardar páginas y documentos de interés para el desarrollo de la aplicación.
* Se ha eso un buen uso de las librerías de Python, siendo usadas prácticamente en casi todas las funcionalidades de la aplicación. Esto ha facilitado mucho la implementación de los aspectos de la aplicación y además ha servido para conocer la dinámica a seguir en la implementación de librerías en Python.
* Por último, se ha conseguido implementar un proceso en el que utilizar un software externo como mini servicio para la aplicación. Esto ha sido un reto que ha llevado mucho tiempo y técnicas que no se esperaban para alcanzar este objetivo.

Como objetivo técnico fallido se podría considerar la implementación de funciones y lógica para permitir la edición de partituras digitalizadas mediante el manejo de la librería Verovio.

**Objetivos personales cumplidos**

Como objetivos personales cumplidos se puede considerar todos cumplidos a excepción del despliegue de la aplicación web a un entorno público que no fue posible debido a que mi aplicación sobrepasaba la memoria permitida dentro del plan gratuito de Heroku.

1. **Líneas de trabajo futuras**

En esta sección se van a detallar y explicar las funcionalidades que no se pudieron alcanzar, así como las implementaciones que ayudarían a que la aplicación esté más completa y funcional.

**Uso de machine learning**

Considerando lo aprendido en el desarrollo de este proyecto y viendo las técnicas y herramientas que utilizan los diferentes softwares relacionados, lo más ideal para la aplicación seria utilizar un modelo entrenado. De esta manera se podría hacer el proceso de transcripción de una manera más efectiva y acorde los intereses del usuario, haciendo que la ratio de éxito al digitalizar cada partitura aumente considerablemente. Habría que buscar la manera idónea para que el modelo se adapte a una colección de partituras dadas y de interés, mejorando la precisión de la inteligencia y con ello del proceso completo.

Durante una de las etapas de investigación de una manera de digitalizar las partituras se valoró usar algún servicio de Google Cloud para el desarrollo del modelo de machine learning. Esta opción resultaba interesante ya que ambos son servicios de Google y están fuertemente integrados en un conjunto de herramientas. Por ello, la implementación de ambos servicios en la aplicación apunta a ser el camino que seguir.

Por ello el desarrollo parece ser, en caso de no encontrar un modelo optimo, entrenar uno acorde a los intereses, para luego implementarlo en la aplicación sustituyéndolo por Audiveris.

**Implementación de una funcionalidad de edición**

Como ya se ha explicado anteriormente, un fin a alcanzar sería que la aplicación diera la opción al usuario de editar y manipular las partituras.

Este objetivo podría ser alcanzado mediante el uso y manipulación de la librería Verovio con funciones. Para ello sería necesario primero documentarse en profundidad de la API y de sus funciones de edición y manipulación. A su vez manejar un renderizado y edición en tiempo real para la implementación de la funcionalidad de edición.

Tras esto sería necesario crear la interfaz de usuario que permita hacer la edición, ya sea agregando notas, eliminándolas, cambiando claves, tempos, etc. La comunicación entre el frontend y el backend en esta funcionalidad sería clave para que los cambios se registren correctamente.

A su vez sería necesario, si se sigue utilizando el servicio, manejar el almacenamiento y recuperación de los archivos con Firebase, siendo necesario adaptar todo el proceso previo a esta herramienta.

Otro aspecto a tener en cuenta son los recursos que consumirá la aplicación tras toda esta implementación ya que el renderizado en tiempo real podría consumir muchos recursos.

**H. Objetivos de Desarrollo Sostenible**

En este apartado se van a listar los objetivos de desarrollo sostenible que alcanza este proyecto.

**Objetivo 4: Educación de calidad**

El proyecto contribuye al Objetivo de Desarrollo Sostenible 4, ‘Educación de calidad’, al aportar en gran medida un recurso funcional para la educación musical. Siendo su acceso libre, permite la edición y manipulación de las partituras en formato digital, abriendo múltiples posibilidades a profesores, alumnos y personas autodidactas para gestionar diferente material del sector.

**Objetivo 9: Industria, innovación e infraestructura**

El proyecto contribuye al Objetivo de Desarrollo Sostenible 9, ‘Industria, innovación e infraestructura’, al implementar una tecnología para la digitalización y edición de partituras, mejorando la infraestructura tecnológica y fomentando la innovación en el sector.

**Objetivo 12: Producción y consumo responsables**

El proyecto contribuye al Objetivo de Desarrollo Sostenible 12, ‘Producción y consumo responsables’, mediante el uso de la herramienta los usuarios reducirán el consumo de papel, al optar por un formato digital. Esta práctica fomenta prácticas más sostenibles que alienta por un recurso más eco-friendly y eficiente.

**Bibliografía**

1. «Optical Music Recognition». En Wikipedia, 12 de febrero de 2024. <https://en.wikipedia.org/w/index.php?title=Optical_music_recognition&oldid=1206438514>.
2. Accedido 27 de junio de 2024. <https://grupo.us.es/gtocoma/pid/pid10/OCR.htm>.
3. perfectovidal. «Binarización por umbral o umbralización». El código Perfecto (blog), 15 de julio de 2015. https://elcodigoperfecto.blog/2015/07/15/binarizacion-por-umbral-o-umbralizacion/.
4. «Eliminación de ruido - MATLAB & Simulink - MathWorks España». Accedido 27 de junio de 2024. <https://es.mathworks.com/help/images/noise-removal.html>.
5. «Filtrado de mediana de 2D - MATLAB medfilt2 - MathWorks España». Accedido 27 de junio de 2024. <https://es.mathworks.com/help/images/ref/medfilt2.html>.
6. «Tipos de operaciones morfológicas - MATLAB & Simulink - MathWorks España». Accedido 27 de junio de 2024. <https://es.mathworks.com/help/images/morphological-dilation-and-erosion.html>.
7. «Tipos de operaciones morfológicas - MATLAB & Simulink - MathWorks España». Accedido 27 de junio de 2024. <https://es.mathworks.com/help/images/morphological-dilation-and-erosion.html>.
8. Pacha, Alexander, Jan Hajič, y Jorge Calvo-Zaragoza. «A Baseline for General Music Object Detection with Deep Learning». *Applied Sciences* 8, n.o 9 (septiembre de 2018): 1488. <https://doi.org/10.3390/app8091488>.
9. «MusicXML». En Wikipedia, la enciclopedia libre, 8 de noviembre de 2023. <https://es.wikipedia.org/w/index.php?title=MusicXML&oldid=155165027>.
10. Delgado-Vázquez, Ángel. «Guías BibUpo: Zotero: Qué es Zotero». Accedido 18 de junio de 2024. <https://guiasbib.upo.es/zotero/que_es_zotero>.
11. «Tool recommendations - Python Packaging User Guide», 18 de junio de 2024..
12. OpenWebinars.net. «Qué es Flask y ventajas que ofrece | OpenWebinars». Accedido 18 de junio de 2024. <https://openwebinars.net/blog/que-es-flask/>.
13. Kinsta®. «Flask vs Django: Let’s Choose Your Next Python Framework», 5 de junio de 2023. <https://kinsta.com/blog/flask-vs-django/>.

The JetBrains Blog. «Django vs Flask: Which Is the Best Python Web Framework? | The PyCharm Blog», 10 de noviembre de 2023. <https://blog.jetbrains.com/pycharm/2023/11/django-vs-flask-which-is-the-best-python-web-framework/>

Django Project. «Django». Accedido 13 de marzo de 2024. <https://www.djangoproject.com/>.

1. Rock Content - ES. «Bootstrap: ¿qué es, para qué sirve y cómo instalarlo?», 12 de abril de 2020. <https://rockcontent.com/es/blog/bootstrap/>.
2. Firebase. «Cloud Firestore | Almacena y sincroniza los datos de tu app a escala global». Accedido 18 de junio de 2024. <https://firebase.google.com/products/firestore?hl=es-419>.

contributors, phpMyAdmin. «phpMyAdmin». phpMyAdmin. Accedido 3 de abril de 2024. <https://www.phpmyadmin.net/>.

«MySQL». Accedido 3 de abril de 2024. <https://www.mysql.com/>.

1. Google Cloud. «Cloud Storage». Accedido 19 de junio de 2024. <https://cloud.google.com/storage>.
2. «Procesamiento de imágenes con OpenCV en Python». Accedido 19 de junio de 2024. <https://imaginaformacion.com/tutoriales/opencv-en-python>.
3. Formación en ciencia de datos | DataScientest.com. «NumPy : La biblioteca de Python más utilizada en Data Science», 18 de enero de 2023. <https://datascientest.com/es/numpy-la-biblioteca-python>.
4. «Audiveris/audiveris». Java. 2016. Reprint, Audiveris, 15 de marzo de 2024. https://github.com/Audiveris/audiveris.
5. Reference book for Verovio. «About This Book», 16 de mayo de 2024. <https://book.verovio.org/introduction/>.

«Optical Music Recognition». En Wikipedia, 12 de febrero de 2024. <https://en.wikipedia.org/w/index.php?title=Optical_music_recognition&oldid=1206438514>.

1. «¿Qué es Docker y cómo funciona? Ventajas de los contenedores Docker». Accedido 19 de junio de 2024. <https://www.redhat.com/es/topics/containers/what-is-docker>.
2. «¿Qué es Heroku? ¿Para qué sirve? ¿Con Salesforce cómo se integra? | SEIDOR». Accedido 18 de junio de 2024. <https://www.seidor.com/blog/heroku-que-es>.
3. Atlassian. «¿Qué es scrum? [+ Cómo empezar]». Atlassian. Accedido 21 de junio de 2024. <https://www.atlassian.com/es/agile/scrum>.
4. contributors, Mark Otto, Jacob Thornton, and Bootstrap. «Introduction». Accedido 21 de junio de 2024. <https://getbootstrap.com/docs/4.1/getting-started/introduction/>.
5. «HTML Basic». Accedido 22 de junio de 2024. https://www.w3schools.com/html/html\_basic.asp.
6. «HTML Examples». Accedido 22 de junio de 2024. <https://www.w3schools.com/html/html_examples.asp>.
7. «Template Designer Documentation — Jinja Documentation (3.1.x)». Accedido 22 de junio de 2024. https://jinja.palletsprojects.com/en/3.1.x/templates/.
8. Python Flask Tutorial #27 🔥 Uploading Images to Firebase with Python 🚀 (Step-by-Step), 2024. <https://www.youtube.com/watch?v=tyF6mVR-ego>.
9. Python Flask Tutorial #29 Python CRUD Operation with Firestore Firebase & Flask, 2024. https://www.youtube.com/watch?v=NAD90zLMRV4.
10. https://community.listopro.com/aprende-a-crear-pantallas-de-login-con-flask-y-firebase/
11. Firebase. «Firebase Admin Python SDK». Accedido 22 de junio de 2024. <https://firebase.google.com/docs/reference/admin/python>.
12. www.javatpoint.com. «Firebase Data Organization in Firestore - Javatpoint». Accedido 26 de junio de 2024. https://www.javatpoint.com/firebase-data-organization-in-firestore.
13. OpenCV. «Home». Accedido 22 de junio de 2024. <https://opencv.org/>.
14. OpenCV. «Get Started». Accedido 22 de junio de 2024. <https://opencv.org/get-started/>.
15. Administrador. ? «? Videostreaming con Flask y OpenCV | Python » omes-va.com». OMES (blog), 16 de diciembre de 2021. <http://omes-va.com/videostreaming-flask-opencv-python/>.
16. «NumPy for MATLAB users — NumPy v2.0 Manual». Accedido 22 de junio de 2024. <https://numpy.org/doc/stable/user/numpy-for-matlab-users.html>.
17. «How to Convert Images to NumPy Array - Javatpoint». Accedido 22 de junio de 2024. <https://www.javatpoint.com/how-to-convert-images-to-numpy-array>.
18. Reference book for Verovio. «Getting Started», 16 de mayo de 2024. <https://book.verovio.org/first-steps/getting-started.html>.
19. «Audiveris/audiveris». Java. 2016. Reprint, Audiveris, 15 de marzo de 2024. <https://github.com/Audiveris/audiveris>.
20. Docker Documentation. «Docker Build», 23:46 + +0100 de 100d. C. <https://docs.docker.com/reference/cli/docker/image/build/>.
21. Cherry Servers. «How to Build a Docker Image from Dockerfile». Accedido 23 de junio de 2024. <https://www.cherryservers.com/blog/docker-build-command>.
22. user7504939. «Answer to “how to run gunicorn on docker”». Stack Overflow, 14 de marzo de 2018. https://stackoverflow.com/a/49287903.
23. «pip freeze - pip documentation v24.1». Accedido 23 de junio de 2024. <https://pip.pypa.io/en/stable/cli/pip_freeze/>.
24. Avid. «PhotoScore & NotateMe Ultimate». Accedido 23 de junio de 2024. https://www.avid.com/products/photoscore-and-notateme-ultimate.