Partituras Musicales. Creación de una aplicación web para validar sistemas de reconocimiento de partituras musicales.(OMR)

Marc Martín Martínez

Índice

1 Introducción **3**

2 Objetivos **3**

3 Planificación **3**
3.1 Introducción al tema: 2 semanas 3
3.2 Análisis de Requisitos: 1 semana 3
3.3 Diseño: 4 semanas 4
3.3.1 Funcionalidad 4
3.3.2 Interfaz de usuario 4
3.4 Implementación y Testing: 6 semanas 4
3.5 Iteraciones: 3 semanas 4

4 Metodología **5**
4.1 Modelo de desarrollo 5
4.2 Herramientas de desarrollo 5
4.2.1 Navegador web 5
4.2.2 Entorno de desarrollo 5
4.2.3 Editor de texto 5

5 Bibliografía **6**

6 Anexo **7**
6.1 Diagrama Gantt 7

1 Introducción

Este proyecto consiste en el diseño de una aplicación web para visualizar partituras musicales y modificarlas. El usuario cargará un archivo MusicXML en la aplicación para después mostrarle la partitura correspondiente. El usuario podrá navegar entre los diferentes compases y modificar tanto las notas y su tempo, entre otros elementos musicales. Una vez el usuario este satisfecho con los cambios, podrá descargar un archivo MusicXML correspondiente a la partitura previamente modificada.

2 Objetivos

El objetivo principal de este proyecto es la creación de un entorno web en el que se puedan visualizar partituras y corregirlas en caso de que sea necesario. Además de también:

- Testear la herramienta con usuarios finales reales con conocimientos musicales.
- Analizar herramientas similares existentes en el mercado.
- Aumentar más los conocimientos sobre la música y su representación.

3 Planificación

En esta sección se representa en diferentes etapas como se desarrollará el proyecto y la duración aproximada de las mismas.
Se usará como herramienta de planificación un diagrama de Gantt para organizar las tareas a hacer y el tiempo que se dispone para ello. Este diagrama puede verse en la figura 6 en la página 10 del documento.
Para la realización de dicho diagrama se utiliza el aplicativo de código abierto *GanttProject*[10].

3.1 Introducción al tema: 2 semanas

Aprendizaje del funcionamiento de MusicXML. [8]
Creación de un glosario para el entendimiento de tecnicismos musicales.
Comprobar el estado del arte. [3] [5] [6] [4] [2] [1]

3.2 Análisis de Requisitos: 1 semana

Identificación de fuentes de requisitos y por ende listar una serie de requisitos.
Clasificación de los requisitos en funcionales, de calidad y, en caso de haberlas, restricciones. Posteriormente, reclasificar estos requisitos siguiendo el modelo de Kano.

3.3 Diseño: 4 semanas

La fase de diseño será dividida en dos subsecciones para distinguir entre el diseño de la funcionalidad de la aplicación y el diseño de la interfaz de usuario.

3.3.1 Funcionalidad

Identificación de patrones ya existentes para su posterior aplicación a la aplicación.
Revisar que se asignan correctamente las responsabilidades correspondientes a cada clase, siguiendo los patrones GRASP.
Creación de diagramas de secuencia y clase.

3.3.2 Interfaz de usuario

Esta fase constará de varias iteraciones en las que el diseño de la UI irá mejorando con cada iteración.

Uso de técnicas y reglas para crear una UI fácil de usar y aprender a la vez que es eficiente y segura.
Crear *personas* para tener una perspectiva mas fiel a la del usuario final de la aplicación.
Crear prototipos de baja fidelidad al inicio para realizar pruebas con posibles usuarios para identificar errores y corregirlos. A medida que hayan mas iteraciones se harán prototipos mas sofisticados y detallados.

3.4 Implementación y Testing: 6 semanas

Estas dos fases normalmente separadas se harán de forma simultanea siguiendo el paradigma *Test-Driven Development*.
Una vez finalizada la implementación se llevaran a cabo pruebas adicionales para asegurar que el programa actúa de forma correcta mediante *exploratory testing*.
Creación de *test cases* y documentarlos de forma adecuada.

3.5 Iteraciones: 3 semanas

Una vez finalizada la implementación y test, se realizarán 3 iteraciones de 1 semana de duración para realizar cambios menores en el proyecto con la ayuda del *feedback* de usuarios finales.

4 Metodología

4.1 Modelo de desarrollo

Inicialmente se seguirá un modelo de desarrollo *waterfall*. Se seguirán las fases de desarrollo del software comunes: análisis de requisitos, diseño, implementación y test. Una vez se haya implementado la visualización y modificación de partituras, se realizará *user testing* en usuarios finales reales. Después de recibir el feedback de dichos tests, el proyecto cambiará a un modelo en espiral, en el que se realizaran hasta 3 iteraciones que consistirán en repetir todas las fases anteriores para realizar cambios menores y realizar otro test con usuarios para la siguiente iteración.

4.2 Herramientas de desarrollo

En esta sección enumeraré las herramientas que se usarán para el proyecto y los motivos de por que se han elegido estas.

4.2.1 Navegador web

Hay 2 opciones claras: *Google Chrome* y *Firefox Developer Edition*. Después de buscar información sobre cual podría ser más útil u óptimo [7] [9] llegué a la conclusión de que utilizaría Google Chrome, ya que es mas popular (para los usuarios finales) y ya tengo experiencia con él.

4.2.2 Entorno de desarrollo

En este caso se usará *Code* de Microsoft ya que es muy popular y cuenta con muchos plugins que pueden ser útiles en el desarrollo de la app.

4.2.3 Editor de texto

Usaré *Overleaf*, un editor de LaTeX online ya que presta muchas funcionalidades para crear informes claros y estéticos. Además de que cuenta con varios paquetes como *bibtex*, una herramienta muy útil para la gestión de bibliografía.

5 Bibliografía

[1] Arnau Baro-Mas. *Optical Music Recognition by Long Short-Term Memory Recurrent Neural Networks*. 2017.

[2] Arnau Baro-Mas. *Recognition of handwritten music scores*. 2016.

[3] Arnau Baró et al. "From Optical Music Recognition to Handwritten Music Recognition: A baseline". In: *Patter Recognition Letters* 123 (2019), pp. 1–8. DOI: https://doi.org/10.1016/j.patrec.2019.02.029.

[4] Jorge Calvo-Zaragoza, Jan Hajič Jr., and Alexander Pacha. *Understanding Optical Music Recognition*. 2019. arXiv: 1908.03608 [cs.CV].

[5] *flat.io*. URL: https://flat.io.

[6] Heinzr. *Assessments for slurs/ties and for articulations*. URL: https://omr-research.net/2019/03/04/assessments-for-slurs-ties-and-for-articulations/.

[7] B.J. Keeton. *Firefox Developer Edition: Can It Replace Google Chrome?* URL: https://www.elegantthemes.com/blog/resources/firefox-developer-edition-can-it-replace-google-chrome.

[8] Make Music. *User manuals for MusicXML*. URL: http://usermanuals.musicxml.com/MusicXML/MusicXML.htm.

[9] Noupe Editorial Team. *Firefox Developer Edition: The Browser for Developers*. URL: https://www.noupe.com/design/firefox-developer-edition-the-browser-for-developers-94805.html.

[10] Alexandre Thomas and Dmitry Barashev. *GanttProject*. URL: https://www.ganttproject.biz/.

6 Anexo

6.1 Diagrama Gantt

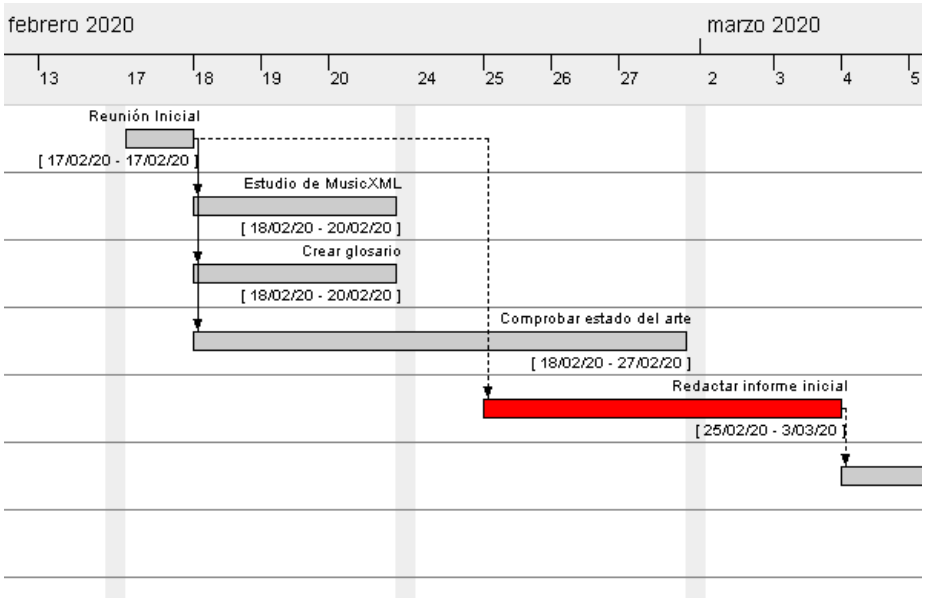

Fig. 1: Inicio del proyecto. Introducción al tema.

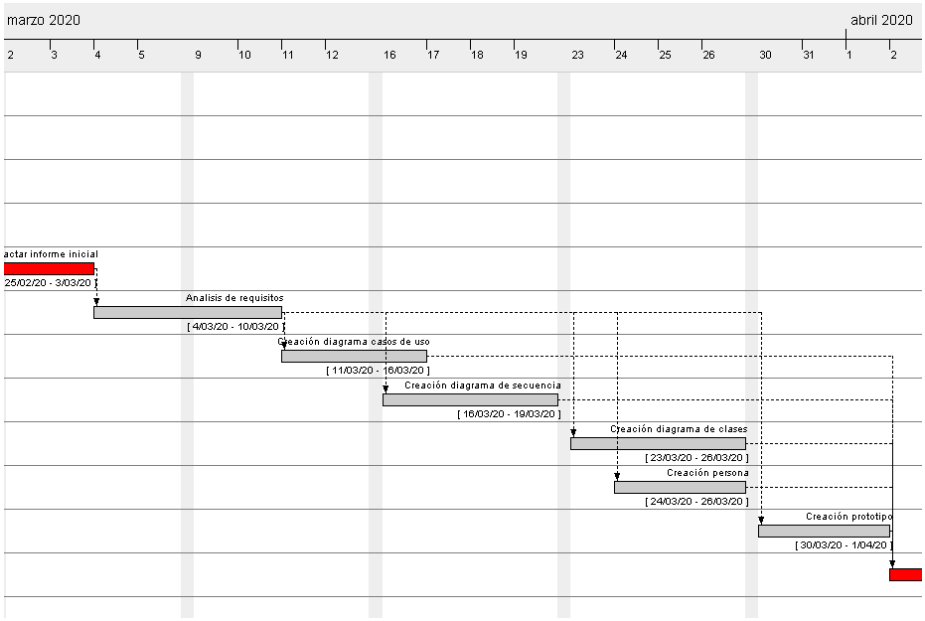

Fig. 2: Análisis de requisitos y diseño

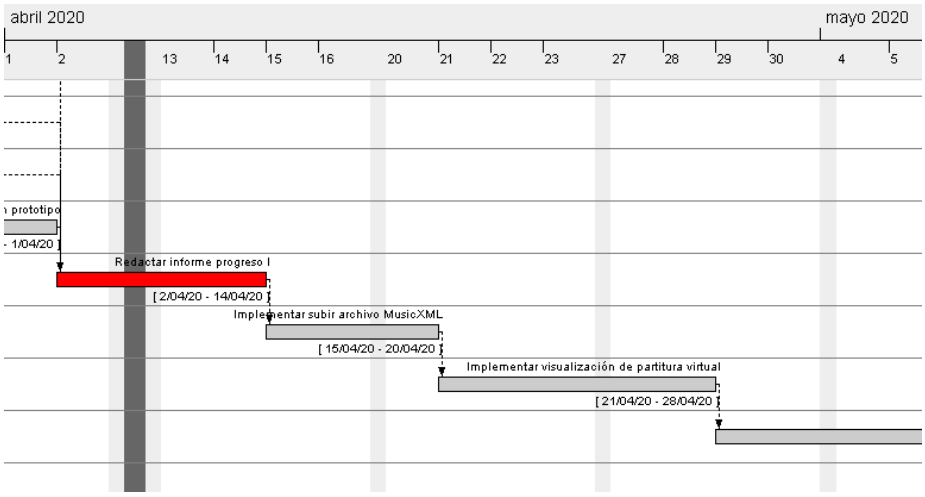

Fig. 3: Implementación y testing I.

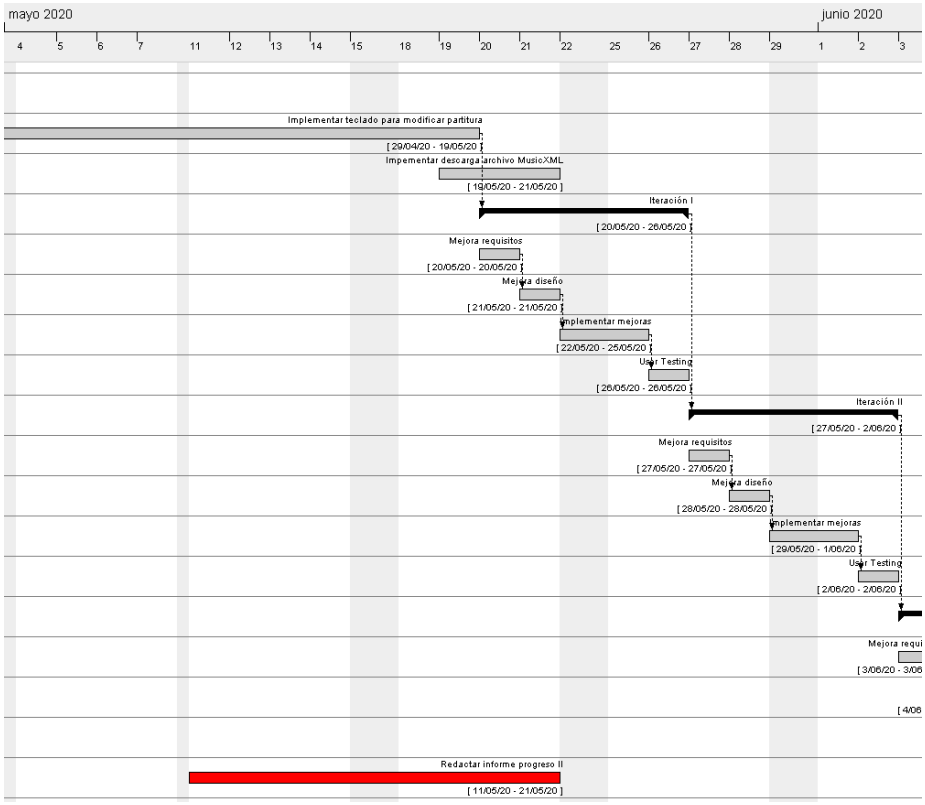

Fig. 4: Implementación y testing II e iteraciones.

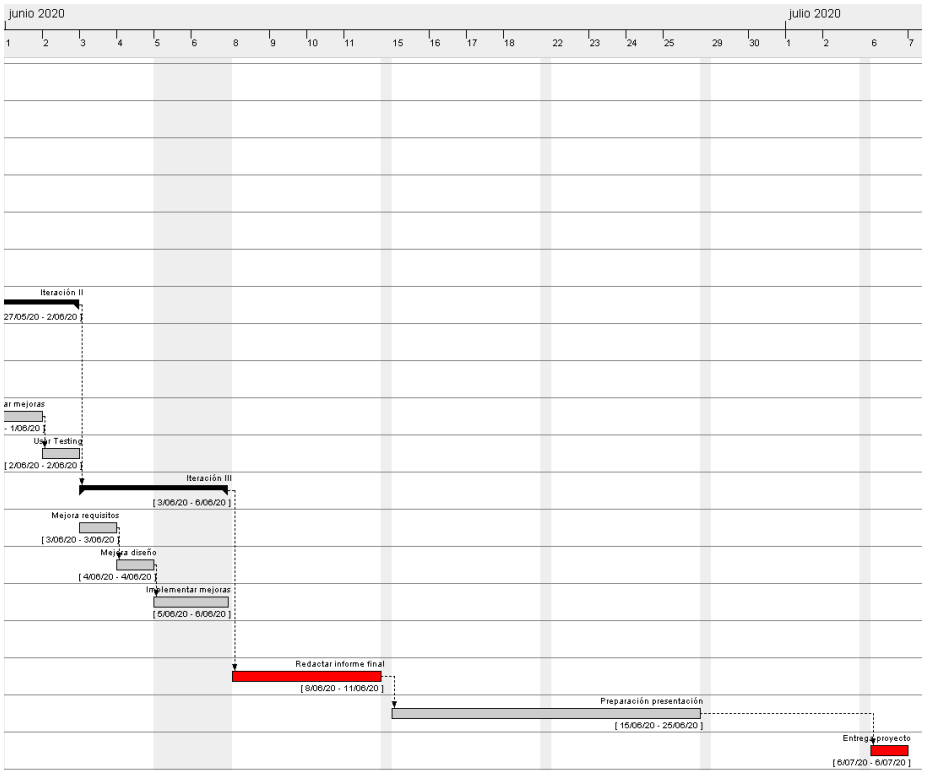

Fig. 5: Final del proyecto.

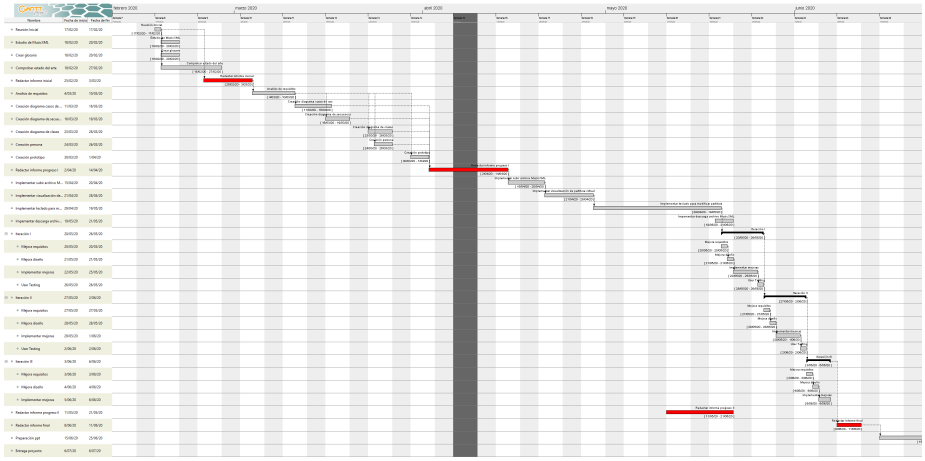

Fig. 6: Diagrama de Gantt.